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

Z. Du
P. Liu
L. Geng
China Mobile

November 2, 2020

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

Abstract

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

Requirements Language

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

Status of This Memo

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 May 6, 2021.

Copyright Notice

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

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect

to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Problem Statement 2
2. Mechanism to Decrease Micro-bursts 3
2.1. Process of Edge Node 3
2.2. Process of Forwarding Node 4
3. IANA Considerations 4
4. Security Considerations 4
5. Acknowledgements 4
6. References 5
6.1. Normative References 5
6.2. Informative References 5
Authors' Addresses 5

1. Problem Statement

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

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

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

The current forwarding mechanism in an IP router is based on statistical multiplexing, and cannot provide the deterministic

service because of various reasons. Even be given a high priority, a deterministic packet can experience a long congestion delay or be lost in a relatively light-loaded network, which is called micro-burst in the network.

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

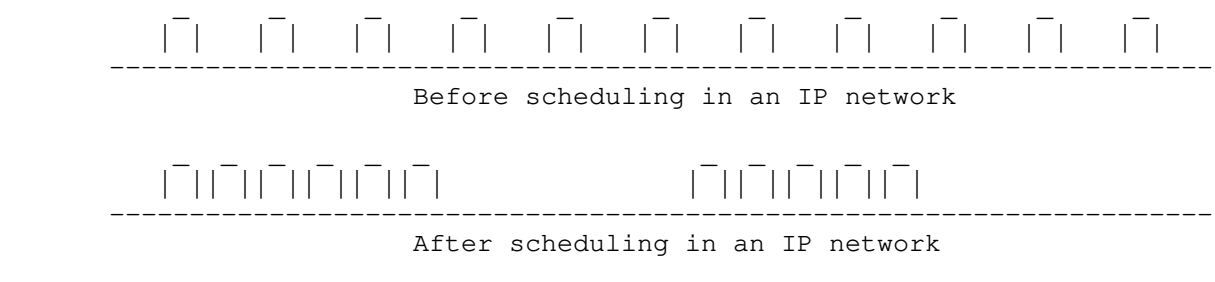


Figure 1: Change of the traffic characteristics in an IP network

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

2. Mechanism to Decrease Micro-bursts

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

2.1. Process of Edge Node

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

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

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

2.2. Process of Forwarding Node

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

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

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

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

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

3. IANA Considerations

TBD.

4. Security Considerations

TBD.

5. Acknowledgements

TBD.

6. References

6.1. Normative References

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

6.2. Informative References

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

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

Authors' Addresses

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

Email: duzongpeng@foxmail.com

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

Email: liupengyjy@chinamobile.com

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

Email: gengliang@chinamobile.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: January 8, 2023

Z. Du
P. Liu
China Mobile
July 7, 2022

Micro-burst Decreasing in Layer3 Network for Low-Latency Traffic
draft-du-detnet-layer3-low-latency-05

Abstract

It is complex to support deterministic forwarding in a large scale network because there is too much dynamic traffic in the network and the data model becomes hard to predict after traffic aggregation on the intermediate nodes. This document introduces the problem of micro-bursts in the layer3 network, and analyses the 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 January 8, 2023.

Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents

(https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction 2
2. Gaps for Large-scale Layer 3 Deterministic Network 3
3. Micro-burst Problem in IP Forwarding 3
4. Analysis of the Method to Decrease Micro-bursts 5
5. An Example of Method to Decrease Micro-bursts 5
5.1. Working Flow of the Method 6
5.2. Process of Edge Node 6
5.3. Process of Forwarding Node 6
5.4. Analysis of the Proposed Method 7
6. IANA Considerations 8
7. Security Considerations 8
8. Acknowledgements 8
9. References 8
9.1. Normative References 8
9.2. Informative References 9
Authors' Addresses 10

1. Introduction

The DetNet architecture [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, as mentioned in [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] and [I-D.dang-queuing-with-multiple-cyclic-buffers], which explore the deterministic forwarding over a large-scale network.

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

2. Gaps for Large-scale Layer 3 Deterministic Network

In this document, the large-scale network means that there are many dynamic flows in the network, but it is hard to do per-flow shaping on the intermediate nodes because they have high pressure on forwarding on the data plane.

According to [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 long 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. An analysis job about the requirements of the large scaled DetNet network is being done in [I-D.liu-detnet-large-scale-requirements].

3. Micro-burst Problem in IP Forwarding

The current IP forwarding mechanism is considered to be a good example fulfilling the requirements of simplicity and scalability. However, the 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 may be unacceptable in the deterministic forwarding. Therefore, they have different design principles in the low layer.

The current forwarding mechanism in an IP router, which is based on statistical multiplexing, can not provide the deterministic service because of various reasons. Even be given a high priority, a critical packet can experience a long congestion delay or be lost in a relatively light-loaded network, which is caused by micro-bursts in the network. The "critical packet" here means that the packet is a DetNet packet, and is sensitive to the latency.

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 (for example, at the granularity of millisecond) or even packet loss.

We introduce the main causes of the micro-burst in the following paragraphs.

Firstly, IP traffic has an 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, or 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.

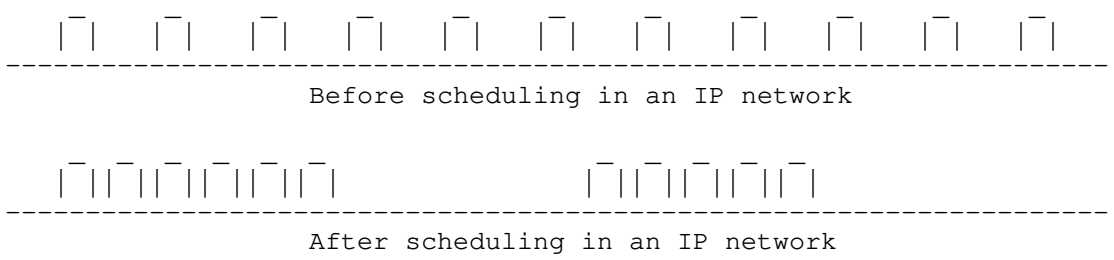


Figure 1: Change of the traffic characteristics in an IP network

4. Analysis of the Method to Decrease Micro-bursts

This document analyses the 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 does not distinguish different critical flows on the forwarding plane on the intermediate nodes.

As talked before, the target method should be scalable and easy to deploy. As the intermediate nodes have high pressure on forwarding packets, the target method should not bring in too much complex process on the data plane. Several requirements are listed as follows.

The first is that the DetNet traffic should support aggregation. The intermediate nodes should not do per-flow process on the date plane.

The second is that separation process of the control plane and data plane on the intermediate nodes. The status of the aggregated DetNet traffic on the control plane may change frequently in the large-scale network. We should not assume that the control plane on an intermediate node can interact with the data plane frequently, for example, to change a shaper parameter frequently. On the data plane, some self-decision process should be supported.

5. An Example of Method to Decrease Micro-bursts

In this section, we describes an example of method fulfilling the requirements mentioned in the last section. It is a traffic forwarding method in the DetNet network, which can decrease micro-bursts for the critical traffic. It needs the cooperation of the edge nodes and the forwarding/intermediate nodes in an IP network.

5.1. Working Flow of the Method

Generally, the method contains two steps:

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

Step2: per interface schedule on the intermediate nodes. 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 or similar to the receiving rate (how to achieve this is out of scope of this document). If all work well, the buffer will be maintained with a proper depth.

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

5.2. 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-burst 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 [IEEE802.1Qav], or the shaping methods in [IEEE802.1Qcr]. 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.

5.3. 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 on the forwarding node. It is to say that, on the forwarding node, the critical flows should be aggregated and handled together.

We do not distinguish each critical flow on the forwarding node, but all the packets of critical flows should have a common identification to be recognisable, which also stands for that the packet is time

sensitive. The forwarding node can obtain the identification in the critical packet, and accordingly forward it to a specific queue.

This document suggests that the forwarding node can deploy a specific queue on each outgoing interface to buffer the time sensitive packets waiting to be sent. When receiving a packet of critical traffic, the forwarding node will forward it to the specific queue on the outgoing interface according to its destination address and its identification. 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.

A 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 uncertainty of micro-bursts.

As no per-flow state is maintained on the forwarding node, the sending rate of the shaper is hard to decide. As said in the last session, the sending rate is suggested to be adjusted referring to the incoming rate of the queue. In other words, before sending the critical packet, we should shape the specific queue by using a shaper parameter based on the computing result of the incoming rate of the queue. The purpose is to maintain a proper buffer depth for the queue.

Although it is claimed that the proposed method is simpler than the TSN mechanisms, forwarding/intermediate nodes also need to be updated. The detailed realization of the method on the intermediate nodes is out of scope of this document.

5.4. Analysis of the Proposed Method

The method proposed does not need synchronization, just as the asynchronous mechanisms studied in [IEEE802.1Qcr]. Furthermore, the method has a larger aggregation granularity, which can fulfill the requirements of simplicity and scalability as much as possible. However, in theory, it has a larger uncertainty on the forwarding than the zero congestion loss target in the TSN mechanisms.

We compare three mechanisms in the following paragraphs. The first is the priority based light-load mechanism, i.e., the traditional

method. The second is the TSN mechanism, such as CQF. The third is the proposed mechanism.

In the first mechanism, we only give a high priority to the critical traffic, and thus the scalability of the deterministic system is good. However, the uncertainty on the forwarding plane perhaps can not fulfill the requirements in the industry network where SLA requirements are very essential. Perhaps, it is only able to work well when a small amount of critical traffic exist in the network.

If we use the scheduling method in the TSN, such as CQF. Its uncertainty is very low, but its scalability is not very good as said in Section 2. It should be noted that in a large deterministic system, the ISP normally will not guarantee the user 100 percent reliability, instead of which it perhaps is a value very close to.

The proposed method has a better scalability than the TSN mechanisms, and a better reliability than the priority based method. If we assume that different services need different deterministic levels, this method may be helpful for the service that does not need a very high deterministic level. For example, the method can be used in the consumption Internet, in which the deterministic service needs a relatively lower deterministic level than the industry Internet.

6. IANA Considerations

This document has no IANA actions.

7. Security Considerations

Detailed security considerations can refer to [I-D.ietf-detnet-bounded-latency] and [I-D.ietf-detnet-security].

8. Acknowledgements

Thanks for the valuable comments from Janos Farkas, Lou Berger, and David Black.

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>.

[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.dang-queuing-with-multiple-cyclic-buffers]
Liu, B. and J. Dang, "A Queuing Mechanism with Multiple Cyclic Buffers", draft-dang-queuing-with-multiple-cyclic-buffers-00 (work in progress), February 2021.

[I-D.ietf-detnet-bounded-latency]
Finn, N., Boudec, J. L., Mohammadpour, E., Zhang, J., and B. Varga, "DetNet Bounded Latency", draft-ietf-detnet-bounded-latency-10 (work in progress), April 2022.

[I-D.ietf-detnet-security]
Grossman, E., Mizrahi, T., and A. J. Hacker, "Deterministic Networking (DetNet) Security Considerations", draft-ietf-detnet-security-16 (work in progress), March 2021.

[I-D.liu-detnet-large-scale-requirements]
Liu, P., Li, Y., Eckert, T., Xiong, Q., and J. Ryoo, "Requirements for Large-Scale Deterministic Networks", draft-liu-detnet-large-scale-requirements-02 (work in progress), April 2022.

[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.

[IEEE802.1Qav]
IEEE 802.1, "IEEE 802.1Qav-2009 - IEEE Standard for Local and metropolitan area networks-- Virtual Bridged Local Area Networks Amendment 12: Forwarding and Queuing Enhancements for Time-Sensitive Streams", 2009, <https://standards.ieee.org/standard/802_1Qav-2009.html>.

[IEEE802.1Qcr]
IEEE 802.1, "IEEE 802.1Qcr-2020 - IEEE Standard for Local and Metropolitan Area Networks--Bridges and Bridged Networks - Amendment 34: Asynchronous Traffic Shaping", 2020, <https://standards.ieee.org/standard/802_1Qcr-2020.html>.

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

[TSN] IEEE 802.1, "Time-Sensitive Networking (TSN) Task Group", 2012, <https://1.ieee802.org/tsn/>.

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

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

X. Geng
M. Chen
F. Yang
Huawei Technologies
November 2, 2020

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

Abstract

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

Requirements Language

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

Status of This Memo

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 May 6, 2021.

Copyright Notice

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

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

Table of Contents

1. Introduction 2
2. Terminology and Conventions 3
3. Redundancy Protection in Segment Routing Scenario 3
4. Segment to Support Redundancy Protection 4
4.1. Redundancy Segment 5
4.2. Merging Segment 5
5. Meta Information to Support Redundancy Protection 6
6. Segment Routing Policy to Support Redundancy Protection . . . 6
7. IANA Considerations 6
8. Security Considerations 7
9. Acknowledgements 7
10. References 7
10.1. Normative References 7
10.2. Informative References 7
Authors' Addresses 7

1. Introduction

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

Redundancy Protection is one of the mechanisms to achieve service protection, following the principle of PREOF (Packet Replication/ Elimination/Ordering Function) defined in [RFC8655]. Specifically, replicates the packets of flows into two or more copies, transports

different copies through different path in parallel, eliminates and orders the packets at end to provide redundancy protection.

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

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

2. Terminology and Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [I-D.ietf-spring-srv6-network-programming] and[RFC2119].

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

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

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

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

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

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

3. Redundancy Protection in Segment Routing Scenario

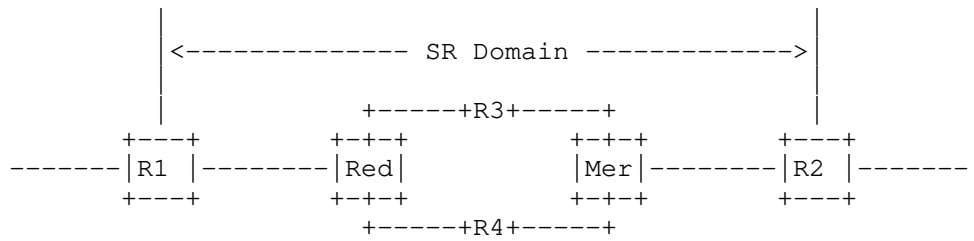


Figure 1: Example Scenario of Redundancy Protection in SR Domain

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

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

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

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

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

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

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

4. Segment to Support Redundancy Protection

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

4.1. Redundancy Segment

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

- Steering the packet into the corresponding redundancy policy
- Packet replication based on the redundancy policy, e.g., the number of replication copies
- Encapsulate the packet with necessary meta data (e.g., flow identification, sequence number) if it is not included in the original packet

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

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

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

4.2. Merging Segment

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

- Packet merging and elimination: forward the first received packets and eliminate the redundant packets
- Packet ordering(optional): reorder the packets if the packet arrives out of order

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

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

```
S01.  When an SRH is processed {
S02.     If (Segments Left>0) & "the packet is not a redundant packet" {
S03.          Do not decrement SL nor update the IPv6 DA with SRH[SL]
S04.          Create a new outer IPv6+SRH-3 header
S05.          Insert the policy-instructed segment lists in the newly created SRH
-3
S06.          Remove the incoming outer IPv6+SRH header
S07.          Set IPv6 DA of IPv6+SRH-3 to the first segment of SRv6 Policy in SR
H-3
S08.     }
S09.     ELSE {
S10.          Drop the packet
S11.     }
```

5. Meta Information to Support Redundancy Protection

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

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

6. Segment Routing Policy to Support Redundancy Protection

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

7. IANA Considerations

This document makes no request of IANA.

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

8. Security Considerations

TBD

9. Acknowledgements

TBD

10. References

10.1. Normative References

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

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

10.2. Informative References

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

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

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

Authors' Addresses

Xuesong Geng
Huawei Technologies

Email: gengxuesong@huawei.com

Mach(Guoyi) Chen
Huawei Technologies

Email: mach.chen@huawei.com

Fan Yang
Huawei Technologies

Email: shirley.yangfan@huawei.com

SPRING Working Group
Internet-Draft
Intended status: Standards Track
Expires: 3 February 2022

X. Geng
M. Chen
F. Yang, Ed.
Huawei Technologies
P. Camarillo
Cisco Systems, Inc.
G. Mishra
Verizon Inc.

2 August 2021

SRv6 for Redundancy Protection
draft-geng-spring-sr-redundancy-protection-05

Abstract

Redundancy Protection is a generalized protection mechanism to achieve the high reliability of service transmission in Segment Routing network. The mechanism inherits the "Live-Live" methodology, targeting to enhance the functionalities of Segment Routing over IPv6. Inspired by DetNet Packet Replication and Packet Elimination functions, two new Segments are introduced to provide replication and elimination functions on specific network nodes by leveraging SRv6 Segment programming capabilities.

Requirements Language

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

Status of This Memo

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 3 February 2022.

Copyright Notice

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

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

Table of Contents

1. Introduction 2
2. Terminology 3
2.1. Requirements Language 3
2.2. Terminology and Conventions 3
3. Redundancy Protection in Segment Routing Scenario 4
4. Segment to Support Redundancy Protection 5
4.1. Redundancy Segment 5
4.2. Merging Segment 6
5. Meta Data to Support Redundancy Protection 7
6. Segment Routing Policy to Support Redundancy Protection . . . 7
7. IANA Considerations 7
8. Security Considerations 8
9. Acknowledgements 8
10. References 8
10.1. Normative References 8
10.2. Informative References 8
Authors' Addresses 8

1. Introduction

Redundancy Protection is a generalized protection mechanism to achieve the high reliability of service transmission in Segment Routing network. Specifically, packets of flows are replicated at a network node into two or more copies, which are transported via different paths in parallel. When copies of packets are received and merged at one network node, the redundant packets are determined and further eliminated to guarantee only one copy of packets is transmitted. The mechanism inherits the "Live-Live" methodology, targeting to enhance the functionalities of Segment Routing over IPv6 [RFC8986]. Inspired by DetNet [RFC8655] Packet Replication and Packet Elimination Functions, two new Segments are introduced to provide the replication and elimination functions on specific network

nodes by leveraging SRv6 Segment programming capabilities. As it is unnecessary to perform switchover between different paths triggered by failure detection, redundancy protection can facilitate to achieve zero packet loss target when failure on either path happens.

Redundancy protection provides ultra reliable protection to many services, for example Cloud VR/Game, IPTV service and other type of video services, high value private line service etc. In this document, redundancy protection is applied to point-to-point service. The mechanism for P2MP service stays out of the scope of this document.

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

This document extends the Segment Routing capabilities to support the redundancy protection in an SRv6 environment, including the definitions of two new Segments, meta data encapsulation, and a variation of Segment Routing Policy.

2. Terminology

2.1. 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.

2.2. Terminology and Conventions

SR: Segment Routing

URLLC: Ultra-Reliable Low-Latency Communication

VR: Virtual Reality

Red Node: Redundancy Node

Mer Node: Merging Node

FID: Flow IDentification

SN: Sequence Number

3. Redundancy Protection in Segment Routing Scenario

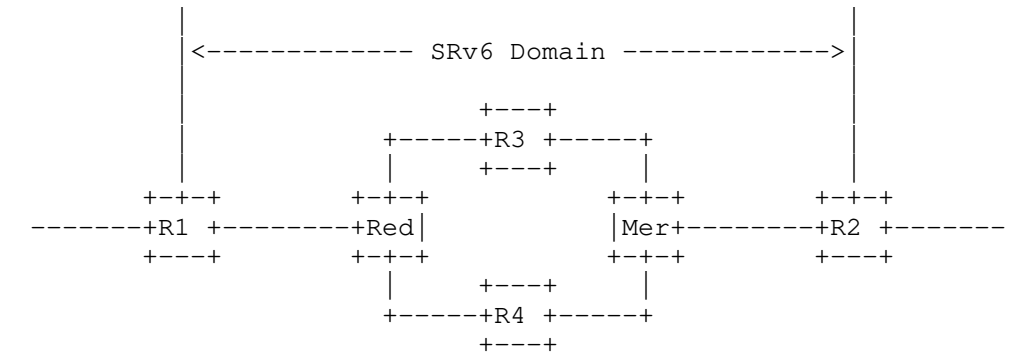


Figure 1: Example Scenario of Redundancy Protection in SRv6 Domain

This figure shows an example of redundancy protection used in SRv6 domain. R1, R2, R3, R4, Red and Mer are SR-capable nodes. When a flow is sent into SRv6 domain, the process is:

1) R1 receives the traffic flow and encapsulates packets with a list of segments destined to R2, which is instantiated as an ordered list of SRv6 SIDs.

2) When the packet flow arrives at Red node, known as Redundancy Node, each packet is replicated into two or more copies. Each copy of the packet is encapsulated with a new segment list, which represents different disjoint forwarding paths.

3) Meta data information such as flow identification (FID) and sequence number (SN) is used to facilitate the packet elimination on Merging node (Mer). Flow identification identifies the specific flow, and sequence number distinguishes the packet sequence of a flow. Meta data is either carried in the packet before it arrives at Red node, or added to each of the replicas at Red node.

4) The multiple replicas go through different paths until the Mer node. The first received packet of the flow is transmitted from Merging Node to R2, and the redundant packets are eliminated.

5) When there is any failures or packet loss in one path, the service continues undisrupted through the other path without break.

6) Sometimes, the packet will arrive out of order because of redundancy protection, the function of reordering may be also necessary on Merging Node. In such case the Merging node may include a reordering function, which is implementation specific and out of the scope of this document.

In this example, service protection is supported by utilizing two packet flows transmitted over two forwarding paths. It is noted that there is no limitation of the number of replicas. For a unidirectional flow, Red node supports replication function, and Mer node supports elimination function. Reordering function MAY be required in combination of elimination function on merging node. To minimize the jitter caused by random packet loss, the disjoint paths are recommended to have similar path forwarding delay.

4. Segment to Support Redundancy Protection

To achieve the packet replication and elimination functions, Redundancy Segment and Merging Segment, as well as the related SRv6 Endpoint Behavior are introduced.

4.1. Redundancy Segment

Redundancy Segment is the identifier of packets which need the replication function on redundancy node. It is also a variation of Binding SID, and associated with a Redundancy Policy to provide segment lists of disjoint paths. Thus, Redundancy segment is associated with service instructions, indicating the following operations:

* Steers the packet into the corresponding redundancy policy

* Encapsulates flow identification and sequence number in packets if the two information is not carried in packets

* Packet replication and segment encapsulation based on the information of redundancy policy, e.g., the number of replication copies, an ordered list of segments with a topological instruction

In the case of SRv6, a new behavior End.R for Redundancy Segment is defined. An instance of a redundancy SID is associated with a redundancy policy B and a source address A. In the following description, End.R behavior is specified in the encapsulation mode. The End.R behavior in the insertion mode is for further study.

When an SRv6-capable node (N) receives an IPv6 packet whose destination address matches a local IPv6 address instantiated as an SRv6 SID (S), and S is a Redundancy SID, N does:

```
S01. When an SRH is processed {
S02.   If (Segments Left>0)   {
S03.     Decrement IPv6 Hop Limit by 1
S04.     Decrement Segments Left by 1
S05.     Update IPv6 DA with Segment List[Segments Left]
S06.     Add flow identification and sequence number if indicated*
S07.     Duplicate the packets (as number of active SID lists in B)
S08.     Push the new IPv6 headers to each replica. The IPv6 header
         contains an SRH with the SID list in B
S09.     Set the outer IPv6 SA to A
S10.     Set the outer IPv6 DA to the first SID of new SRH SL
S11.     Set the outer Payload Length, Traffic Class, Flow Label,
         Hop Limit and Next-Header fields
S12.     Submit the packet to the egress IPv6 FIB lookup
         for transmission to the new destination
S13.   }
S14. }
```

* Adding flow identification and sequence number is an optional behavior for Redundancy Segment. The instruction execution is determined and explicitly indicated by SR policy or Segment itself.

4.2. Merging Segment

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

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

In order to eliminate the redundant packet of a flow, merging node utilizes sequence number to evaluate the redundant status of a packet. Note that implementation specific mechanism could be applied to control the amount of state monitored on sequence number, so that system memory usage can be limited at a reasonable level.

As merging node needs to maintain the state of flows, a centralized controller should have a knowledge of merging nodes capability, and never provision the redundancy policy to redundancy node when the computation result goes beyond the flow recovery capability of merging node. The capability advertisement of merging node will be specified separately elsewhere, which is not within the scope of this document.

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

When an SRv6-capable node (N) receives an IPv6 packet whose destination address matches a local IPv6 address instantiated as an SRv6 SID (S), and S is a Merging SID, N does:

```
S01. When an SRH is processed {
S02.  If (Segments Left> or ==0)   {
S03.    Acquire the sequence number of received packet and
        look it up in table
S04.      If (this sequence number does not exist in the table) {
S05.       Store this sequence number in table
S06.       Remove the outer IPv6+SRH header
S07.       Decrement IPv6 Hop Limit by 1 in inner SRH
S08.       Decrement Segments Left by 1 in inner SRH
S09.       Update IPv6 DA with Segment List[Segments Left] in inner SRH
S10.       Submit the packet to the egress IPv6 FIB lookup and transmit
S11.      }
S12.      ELSE {
S13.           Drop the packet
S14.      }
S15.    }
S16. }
```

5. Meta Data to Support Redundancy Protection

To support the redundancy protection function, flow identification and sequence number are required. Flow identification identifies one specific flow of redundancy protection, and is usually allocated from centralized controller to the SR ingress node or redundancy node in SR network. Sequence number distinguishes the packets within a flow by specifying the order of packets. It is usually generated at SR ingress node. If necessary, redundancy node can also facilitate to add sequence number if required. Thus, encapsulations of flow identification and sequence number should be specified accordingly.

6. Segment Routing Policy to Support Redundancy Protection

Redundancy Policy is a variation of SR Policy to conduct the replicas to multiple disjoint paths for redundancy protection. It extends SR policy to include more than one ordered lists of segments between redundancy node and merging node, and all the ordered lists of segments are used at the same time to steer the copies of flow into different disjoint paths.

7. IANA Considerations

This document requires registration of End.R behavior and End.M behavior in "SRv6 Endpoint Behaviors" sub-registry of "Segment Routing Parameters" registry.

8. Security Considerations

TBD

9. Acknowledgements

The authors would like to thank Bruno Decraene, Ron Bonica, James Guichard, Jeffrey Zhang, Balazs Varga for their valuable comments and discussions.

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>.

[RFC8986] Filsfils, C., Ed., Camarillo, P., Ed., Leddy, J., Voyer, D., Matsushima, S., and Z. Li, "Segment Routing over IPv6 (SRv6) Network Programming", RFC 8986, DOI 10.17487/RFC8986, February 2021, <https://www.rfc-editor.org/info/rfc8986>.

10.2. Informative References

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

Authors' Addresses

Xuesong Geng
Huawei Technologies
China

Email: gengxuesong@huawei.com

Mach(Guoyi) Chen
Huawei Technologies
China

Email: mach.chen@huawei.com

Fan Yang
Huawei Technologies
China

Email: shirley.yangfan@huawei.com

Pablo Camarillo Garvia
Cisco Systems, Inc.
Spain

Email: pcamaril@cisco.com

Gyan Mishra
Verizon Inc.

Email: gyan.s.mishra@verizon.com

DetNet
Internet-Draft
Intended status: Informational
Expires: May 6, 2021

N. Finn
Huawei Technologies Co. Ltd
J-Y. Le Boudec
E. Mohammadpour
EPFL
J. Zhang
Huawei Technologies Co. Ltd
B. Varga
J. Farkas
Ericsson

November 2, 2020

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

Abstract

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

Status of This Memo

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 May 6, 2021.

Copyright Notice

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

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

Table of Contents

1. Introduction 2
2. Terminology and Definitions 3
3. DetNet bounded latency model 3
3.1. Flow creation 4
3.1.1. Static flow latency calculation 4
3.1.2. Dynamic flow latency calculation 5
3.2. Relay node model 6
4. Computing End-to-end Delay Bounds 8
4.1. Non-queuing delay bound 8
4.2. Queuing delay bound 8
4.2.1. Per-flow queuing mechanisms 9
4.2.2. Per-class queuing mechanisms 9
4.3. Ingress considerations 10
4.4. Interspersed non-DetNet transit nodes 11
5. Achieving zero congestion loss 11
6. Queuing techniques 12
6.1. Queuing data model 12
6.2. Preemption 14
6.3. Time Aware Shaper 15
6.4. Credit-Based Shaper with Asynchronous Traffic Shaping . . 15
6.4.1. Delay Bound Calculation 17
6.4.2. Flow Admission 18
6.5. IntServ 19
6.6. Cyclic Queuing and Forwarding 20
7. References 21
7.1. Normative References 21
7.2. Informative References 22
Authors' Addresses 23

1. Introduction

The ability for IETF Deterministic Networking (DetNet) or IEEE 802.1 Time-Sensitive Networking (TSN, [IEEE8021TSN]) to provide the DetNet services of bounded latency and zero congestion loss depends upon A) configuring and allocating network resources for the exclusive use of DetNet/TSN flows; B) identifying, in the data plane, the resources to

be utilized by any given packet, and C) the detailed behavior of those resources, especially transmission queue selection, so that latency bounds can be reliably assured. Thus, DetNet is an example of an IntServ Guaranteed Quality of Service [RFC2212]

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

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

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

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

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

2. Terminology and Definitions

This document uses the terms defined in [RFC8655].

3. DetNet bounded latency model

3.1. Flow creation

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

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

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

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

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

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

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

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

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

3.1.1. Static flow latency calculation

The static problem:

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

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

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

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

3.1.2. Dynamic flow latency calculation

The dynamic problem:

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

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

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

3.2. Relay node model

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

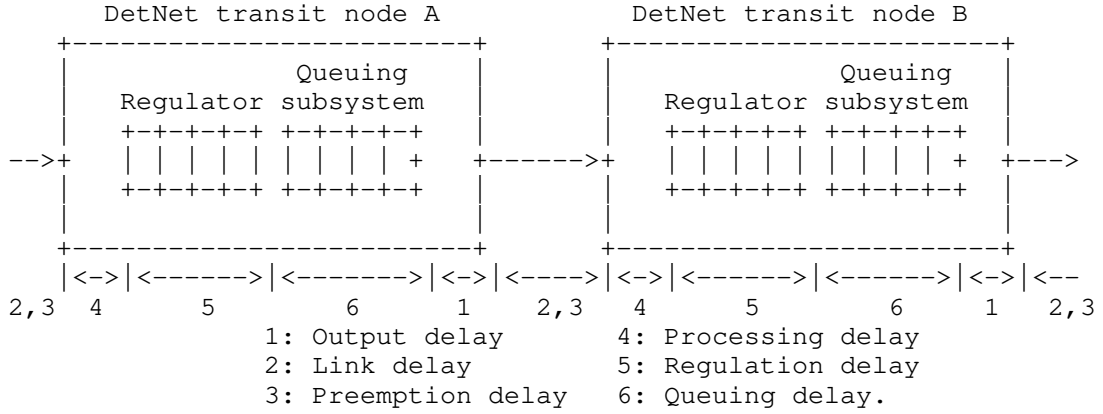


Figure 1: Timing model for DetNet or TSN

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

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

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

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

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

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

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

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

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

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

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

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

4. Computing End-to-end Delay Bounds

4.1. Non-queuing delay bound

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

```
end_to_end_delay_bound = non_queuing_delay_bound +
queuing_delay_bound
```

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

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

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

4.2. Queuing delay bound

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

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

4.2.1. Per-flow queuing mechanisms

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

4.2.2. Per-class queuing mechanisms

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

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

Figure 2 shows an example of a network with 5 nodes, per-class queuing mechanism and interleaved regulators as in Figure 1. An end-

to-end delay bound for flow f, traversing nodes 1 to 5, is calculated as follows:

end_to_end_latency_bound_of_flow_f = C12 + C23 + C34 + S4

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

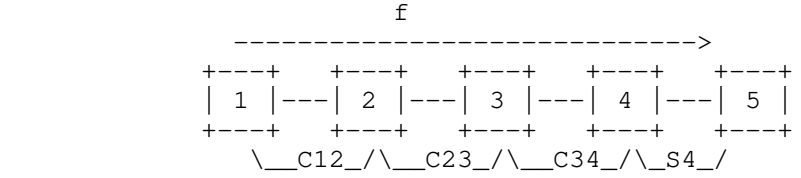


Figure 2: End-to-end delay computation example

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

4.3. Ingress considerations

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

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

4.4. Interspersed non-DetNet transit nodes

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

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

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

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

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

5. Achieving zero congestion loss

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

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

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

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

Let

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

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

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

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

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

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

6. Queuing techniques

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

6.1. Queuing data model

Sophisticated queuing mechanisms are available in Layer 3 (L3, see, e.g., [RFC7806] for an overview). In general, we assume that "Layer 3" queues, shapers, meters, etc., are precisely the "regulators" shown in Figure 1. The "queuing subsystems" in this figure are not the province solely of bridges; they are an essential part of any DetNet transit node. As illustrated by numerous implementation examples, some of the "Layer 3" mechanisms described in documents such as [RFC7806] are often integrated, in an implementation, with

the "Layer 2" mechanisms also implemented in the same node. An integrated model is needed in order to successfully predict the interactions among the different queuing mechanisms needed in a network carrying both DetNet flows and non-DetNet flows.

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

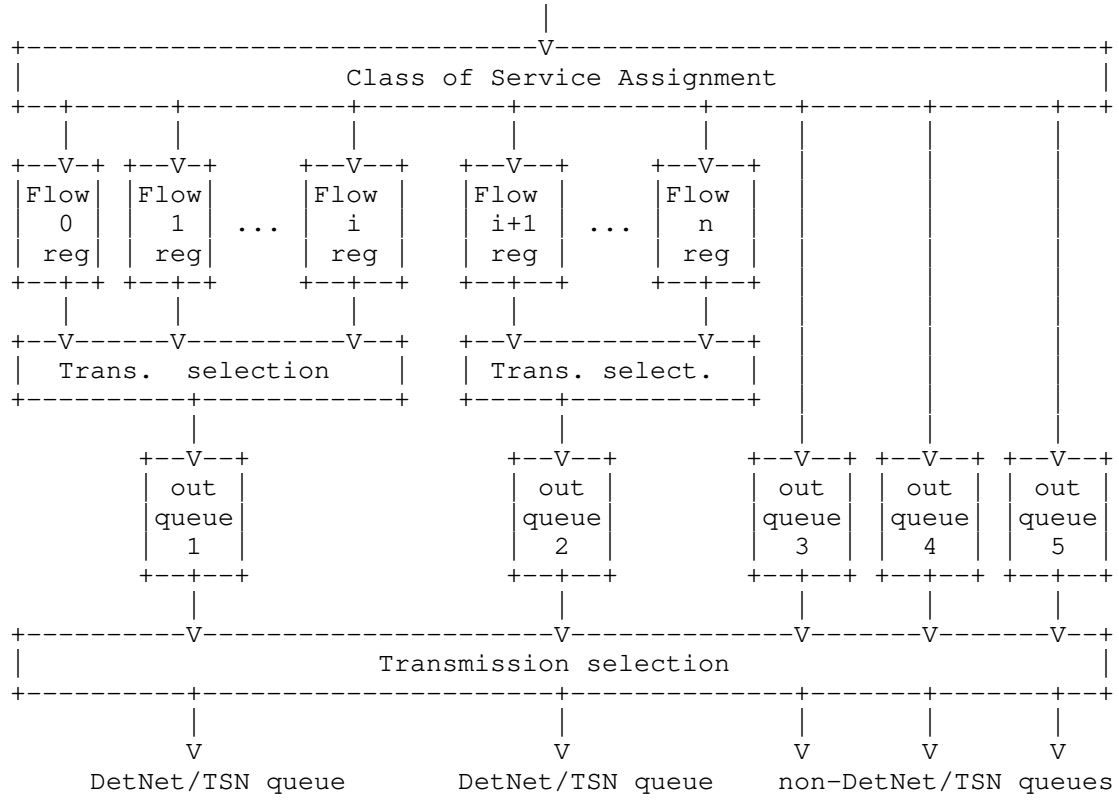


Figure 3: IEEE 802.1Q Queuing Model: Data flow

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

- Discarding packets because a queue is full.

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

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

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

- Input port.

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

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

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

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

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

6.2. Preemption

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

in DetNet nodes. Best-effort queues pass through the interruptible MAC, and can thus be preempted.

6.3. Time Aware Shaper

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

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

6.4. Credit-Based Shaper with Asynchronous Traffic Shaping

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

Based on the timing model described in Figure 1, the contention occurs only at the output port of a relay node; therefore, the focus of the rest of this subsection is on the regulator and queuing subsystem in the output port of a relay node. The output port performs per-class scheduling with eight classes (queuing subsystems): one for CDT, one for class A traffic, one for class B traffic, and five for BE traffic denoted as BE0-BE4. The queuing policy for each queuing subsystem is FIFO. In addition, each node output port also performs per-flow regulation for AVB flows using an interleaved regulator (IR), called Asynchronous Traffic Shaper [IEEE8021Qcr]. Thus, at each output port of a node, there is one interleaved regulator per-input port and per-class; the interleaved regulator is mapped to the regulator depicted in Figure 1. The detailed picture of scheduling and regulation architecture at a node output port is given by Figure 4. The packets received at a node

input port for a given class are enqueued in the respective interleaved regulator at the output port. Then, the packets from all the flows, including CDT and BE flows, are enqueued in queuing subsytem; there is no regulator for such classes.

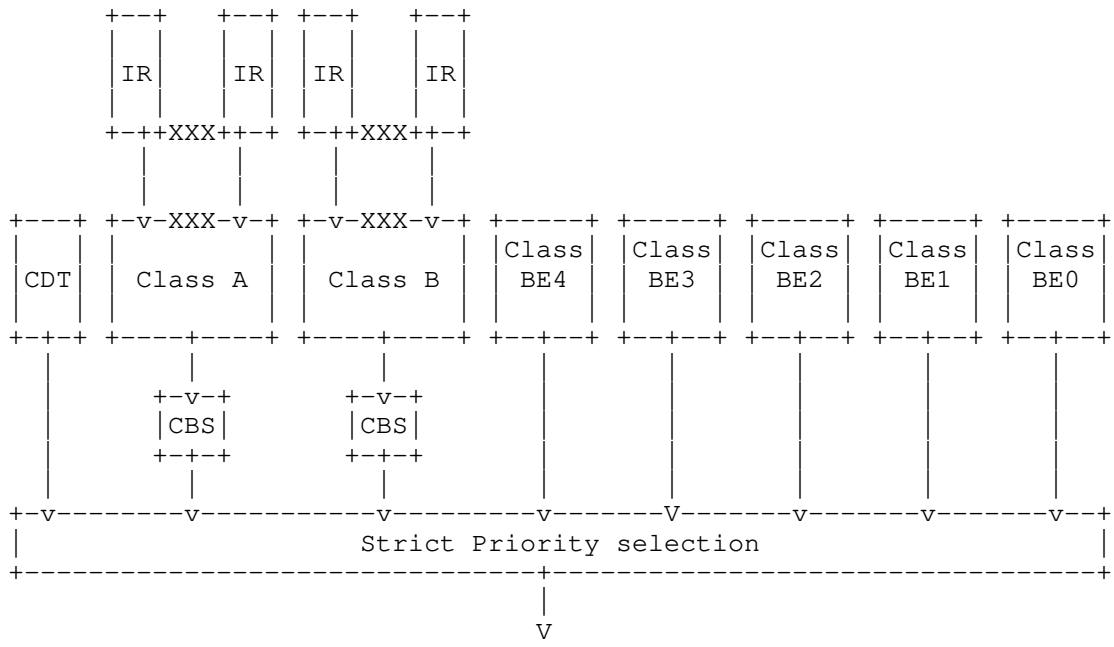


Figure 4: The architecture of an output port inside a relay node with interleaved regulators (IRs) and credit-based shaper (CBS)

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

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

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

6.4.1. Delay Bound Calculation

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

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

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

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

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

If the flow is of class A:

$$R_A = I_A (c - r_h)/ c$$

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

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

If the flow is of class B:

$$R_B = I_B (c - r_h)/ c$$

$$T_B = (L_BE + L_A + L_nA I_A/(c_h - I_A) + b_h + r_h L_n/c)/(c - r_h)$$

where L_A is the maximum packet length of class A; L_BE is the maximum packet length of class BE.

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

Cij = d_X

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

6.4.2. Flow Admission

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

The static admission control:

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

The dynamic admission control:

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

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

R_acc <=R, (Eq. 1)

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

b_acc <=b_t. (Eq. 2)

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

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

6.5. IntServ

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

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

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

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

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

6.6. Cyclic Queuing and Forwarding

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

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

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

(h+1) T_c

and the minimum delay is:

(h-1) T_c

which gives a latency variation of 2T_c.

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

cycle. A two-buffer CQF is recommended if DT is small compared to T_c. For a large DT, CQF with more buffers can be used.

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

7. References

7.1. Normative References

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

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

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

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

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

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

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

7.2. Informative References

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

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

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

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

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

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

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

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

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

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

Authors' Addresses

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

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

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

Email: jean-yves.leboudec@epfl.ch

Ehsan Mohammadpour
EPFL
IC Station 14
Lausanne EPFL 1015
Switzerland

Email: ehsan.mohammadpour@epfl.ch

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

Email: zhangjiayi11@huawei.com

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

Email: balazs.a.varga@ericsson.com

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

Email: janos.farkas@ericsson.com

DetNet
Internet-Draft
Intended status: Informational
Expires: 10 October 2022

N. Finn
Huawei Technologies Co. Ltd
J-Y. Le Boudec
E. Mohammadpour
EPFL
J. Zhang
Huawei Technologies Co. Ltd
B. Varga
Ericsson

8 April 2022

DetNet Bounded Latency
draft-ietf-detnet-bounded-latency-10

Abstract

This document presents a timing model for sources, destinations, and DetNet transit nodes. Using the model, it provides a methodology to compute end-to-end latency and backlog bounds for various queuing methods. The methodology can be used by the management and control planes and by resource reservation algorithms to provide bounded latency and zero congestion loss for the DetNet service.

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 10 October 2022.

Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction 3
2. Terminology and Definitions 4
3. DetNet bounded latency model 4
3.1. Flow admission 4
3.1.1. Static latency-calculation 5
3.1.2. Dynamic latency-calculation 6
3.2. Relay node model 7
4. Computing End-to-end Delay Bounds 9
4.1. Non-queuing delay bound 9
4.2. Queuing delay bound 10
4.2.1. Per-flow queuing mechanisms 11
4.2.2. Aggregate queuing mechanisms 11
4.3. Ingress considerations 12
4.4. Interspersed DetNet-unaware transit nodes 13
5. Achieving zero congestion loss 13
6. Queuing techniques 14
6.1. Queuing data model 15
6.2. Frame Preemption 17
6.3. Time-Aware Shaper 17
6.4. Credit-Based Shaper with Asynchronous Traffic Shaping . . 18
6.4.1. Delay Bound Calculation 20
6.4.2. Flow Admission 21
6.5. Guaranteed-Service IntServ 22
6.6. Cyclic Queuing and Forwarding 23
7. Example application on DetNet IP network 24
8. Security considerations 26
9. IANA considerations 27
10. Acknowledgement 27
11. Contributors 27
12. References 27
12.1. Normative References 27
12.2. Informative References 28
Authors' Addresses 30

1. Introduction

The ability for IETF Deterministic Networking (DetNet) or IEEE 802.1 Time-Sensitive Networking [IEEE8021TSN] to provide the DetNet services of bounded latency and zero congestion loss depends upon

A) configuring and allocating network resources for the exclusive use of DetNet flows;

B) identifying, in the data plane, the resources to be utilized by any given packet;

C) the detailed behavior of those resources, especially transmission queue selection, so that latency bounds can be reliably assured.

As explained in [RFC8655], DetNet flows are notably characterized by

1. a maximum bandwidth, guaranteed either by the transmitter or by strict input metering;

2. a requirement for a guaranteed worst-case end-to-end latency.

That latency guarantee, in turn, provides the opportunity for the network to supply enough buffer space to guarantee zero congestion loss. It is assumed in this document that the paths of DetNet flows are fixed. Before the transmission of a DetNet flow, it is possible to calculate end-to-end latency bounds and the amount of buffer space required at each hop to ensure zero congestion loss; this can be used by the applications identified in [RFC8578].

This document presents a timing model for sources, destinations, and the DetNet transit nodes; using this model, it provides a methodology to compute end-to-end latency and backlog bounds for various queuing mechanisms that can be used by the management and control planes to provide DetNet qualities of service. The methodology used in this document account for the possibility of packet reordering within a DetNet node. The bounds on the amount of packet reordering is out of the scope of this document and can be found in [PacketReorderingBounds]. Moreover, this document references specific queuing mechanisms, mentioned in [RFC8655], as proofs of concept that can be used to control packet transmission at each output port and achieve the DetNet quality of service.

Using the model presented in this document, it is possible for an implementer, user, or standards development organization to select a set of queuing mechanisms for each device in a DetNet network, and to select a resource reservation algorithm for that network, so that

those elements can work together to provide the DetNet service. Section 7 provides an example application of the timing model introduced in this document on a DetNet IP network with a combination of different queuing mechanisms.

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

2. Terminology and Definitions

This document uses the terms defined in [RFC8655]. Moreover, the following terms are used in this document:

T-SPEC
: TrafficSpecification as defined in Section 5.5 of [RFC9016].

arrival curve
: An arrival curve function alpha(t) is an upper bound on the number of bits seen at an observation point within any time interval t.

CQF
: Cyclic Queuing and Forwarding.

CBS
: Credit-based Shaper.

TSN
: Time-Sensitive Networking.

PREOF
: A collective name for Packet Replication, Elimination, and Ordering Functions.

Packet Ordering Function (POF)
: A function that reorders packets within a DetNet flow that are received out of order. This function can be implemented by a DetNet edge node, a DetNet relay node, or an end system.

3. DetNet bounded latency model

3.1. Flow admission

This document assumes that the following paradigm is used to admit DetNet flows:

1. Perform any configuration required by the DetNet transit nodes in the network for aggregates of DetNet flows. This configuration is done beforehand, and not tied to any particular DetNet flow.

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

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

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

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

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

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

3.1.1. Static latency-calculation

The static problem:

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

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

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

3.1.2. Dynamic latency-calculation

The dynamic problem:

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

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

Dynamic latency-calculation can be done based on the static one described in Section 3.1.1; when a new DetNet flow is created or deleted, the entire calculation for all DetNet flows is repeated. If an already-established DetNet flow would be pushed beyond its latency requirements by the new DetNet flow request, then the new DetNet flow request can be refused, or some other suitable action taken.

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

3.2. Relay node model

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

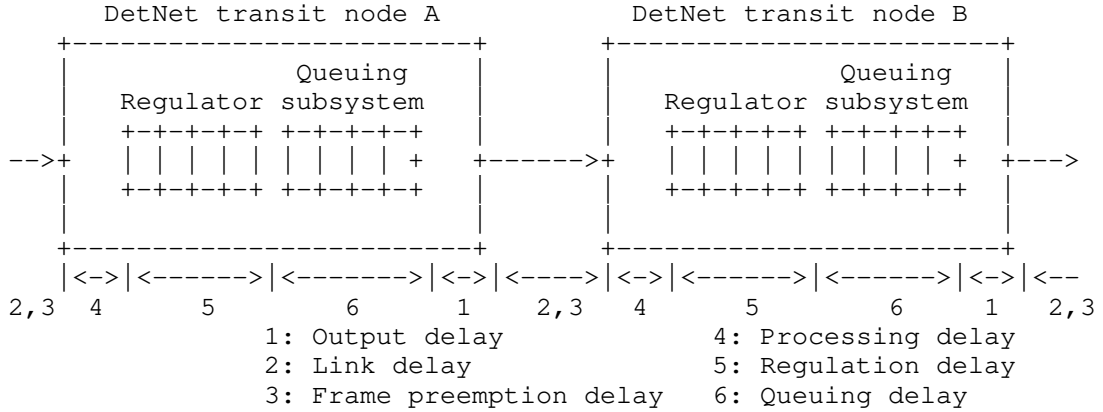


Figure 1: Timing model for DetNet or TSN

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

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

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

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

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

4. Processing delay
   This delay covers the time from the reception of the last bit of the packet to the time the packet is enqueued in the regulator (queuing subsystem, if there is no regulator) as shown in Figure 1. This delay can be variable, and depends on the details of the operation of the forwarding node.

5. Regulator delay
   A regulator, also known as shaper in [RFC2475], delays some or all of the packets in a traffic stream in order to bring the stream into compliance with an arrival curve; an arrival curve 'alpha(t)' is an upper bound on the number of bits observed within any interval t. The regulator delay is the time spent from the insertion of the last bit of a packet into a regulation queue until the time the packet is declared eligible according to its regulation constraints. We assume that this time can be calculated based on the details of regulation policy. If there is no regulation, this time is zero.

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

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

In this analysis, the measurement is from the point at which a packet is selected for output in a node to the point at which it is selected for output in the next downstream node (that is the definition of a "hop"). In general, any queue selection method that is suitable for use in a DetNet network includes a detailed specification as to exactly when packets are selected for transmission. Any variations

in any of the delay times 1-4 result in a need for additional buffers in the queue. If all delays 1-4 are constant, then any variation in the time at which packets are inserted into a queue depends entirely on the timing of packet selection in the previous node. If the delays 1-4 are not constant, then additional buffers are required in the queue to absorb these variations. Thus:

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

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

4. Computing End-to-end Delay Bounds

4.1. Non-queuing delay bound

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

```
end_to_end_delay_bound = non_queuing_delay_bound +
queuing_delay_bound
```

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

First, at the h-th hop along the path of this DetNet flow, obtain an upper-bound per-hop_non_queuing_delay_bound[h] on the sum of the bounds over the delays 1,2,3,4 of Figure 1. These upper bounds are expected to depend on the specific technology of the DetNet transit node at the h-th hop but not on the T-SPEC of this DetNet flow [RFC9016]. Then set non_queuing_delay_bound = the sum of per-hop_non_queuing_delay_bound[h] over all hops h.

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

4.2. Queuing delay bound

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

The computation of per-hop queuing delay bounds must account for the fact that the arrival curve of a DetNet flow is no longer satisfied at the ingress of a hop, since burstiness increases as one flow traverses one DetNet transit node. If a regulator is placed at a hop, an arrival curve of a DetNet flow at the entrance of the queuing subsystem of this hop is the one configured at the regulator (also called shaping curve in [NetCalBook]); otherwise, an arrival curve of the flow can be derived using the delay-jitter of the flow from the last regulation point (the last regulator in the path of the flow if there is any, otherwise the source of the flow) to the ingress of the hop; more formally, assume a DetNet flow has arrival curve at the last regulation point equal to 'alpha(t)', and the delay-jitter from the last regulation point to the ingress of the hop is 'V'. Then, the arrival curve at the ingress of the hop is 'alpha(t+V)'.

For example, consider a DetNet flow with T-SPEC "Interval: tau, MaxPacketsPerInterval: K, MaxPayloadSize: L" at source. Then, a leaky-bucket arrival curve for such flow at source is alpha(t)=r * t+ b, t>0; alpha(0)=0, where r is the rate and b is the bucket size, computed as

r = K * (L+L') / tau,

b = K * (L+L').

where L' is the size of any added networking technology-specific encapsulation (e.g., MPLS label(s), UDP, and IP headers). Now, if the flow has delay-jitter of 'V' from the last regulation point to the ingress of a hop, an arrival curve at this point is r * t + b + r * V, implying that the burstiness is increased by r*V. A more detailed information on arrival curves is available in [NetCalBook].

4.2.1. Per-flow queuing mechanisms

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

4.2.2. Aggregate queuing mechanisms

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

A solution to deal with this issue for the DetNet flows is to reshape them at every hop. This can be done with per-flow regulators (e.g., leaky bucket shapers), but this requires per-flow queuing and defeats the purpose of aggregate queuing. An alternative is the interleaved regulator, which reshapes individual DetNet flows without per-flow queuing ([SpechtUBS], [IEEE8021Qcr]). With an interleaved regulator, the packet at the head of the queue is regulated based on its (flow) regulation constraints; it is released at the earliest time at which this is possible without violating the constraint. One key feature of per-flow or interleaved regulator is that, it does not increase worst-case latency bounds [LeBoudecTheory]. Specifically, when an interleaved regulator is appended to a FIFO subsystem, it does not increase the worst-case delay of the latter; in Figure 1, when the order of packets from output of queuing subsystem at node A to the

entrance of regulator at node B is preserved, then the regulator does not increase the worst-case latency bounds; this is made possible if all the systems are FIFO or a DetNet packet-ordering function (POF) is implemented just before the regulator. This property does not hold if packet reordering occurs from the output of a queuing subsystem to the entrance of next downstream interleaved regulator, e.g., at a non-FIFO switching fabric.

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

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

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

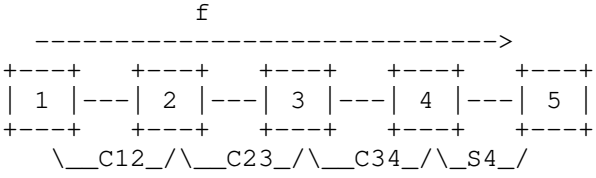


Figure 2: End-to-end delay computation example

REMARK: If packet reordering does not occur, the end-to-end latency bound calculation provided here gives a tighter latency upper-bound than would be obtained by adding the latency bounds of each node in the path of a DetNet flow [TSNwithATS].

4.3. Ingress considerations

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

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

4.4. Interspersed DetNet-unaware transit nodes

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

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

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

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

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

5. Achieving zero congestion loss

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

To avoid congestion losses, an upper bound on the backlog present in the regulator and queuing subsystem of Figure 1 must be computed during resource reservation. This bound depends on the set of flows

that use these queues, the details of the specific queuing mechanism and an upper bound on the processing delay (4). The queue must contain the packet in transmission plus all other packets that are waiting to be selected for output. A conservative backlog bound, that applies to all systems, can be derived as follows.

The backlog bound is counted in data units (bytes, or words of multiple bytes) that are relevant for buffer allocation. For every flow or an aggregate of flows, we need one buffer space for the packet in transmission, plus space for the packets that are waiting to be selected for output.

Let

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

* nb_input_ports be the number input ports that send traffic to this output port

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

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

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

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

The above bound is over the backlog caused by the traffic entering the queue from the input ports of a DetNet node. If the DetNet node also generates packets (e.g., creation of new packets, replication of arriving packets), the bound must accordingly incorporate the introduced backlog.

6. Queuing techniques

In this section, we present a general queuing data model as well as some examples of queuing mechanisms. For simplicity of latency bound computation, we assume leaky-bucket arrival curve for each DetNet flow at source. Also, at each DetNet transit node, the service for each queue is abstracted with a minimum guaranteed rate and a latency [NetCalBook].

6.1. Queuing data model

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

Figure 3 shows the general model for the flow of packets through the queues of a DetNet transit node. The DetNet packets are mapped to a number of regulators. Here, we assume that the PREOF (Packet Replication, Elimination and Ordering Functions) are performed before the DetNet packets enter the regulators. All Packets are assigned to a set of queues. Packets compete for the selection to be passed to queues in the queuing subsystem. Packets again are selected for output from the queuing subsystem.

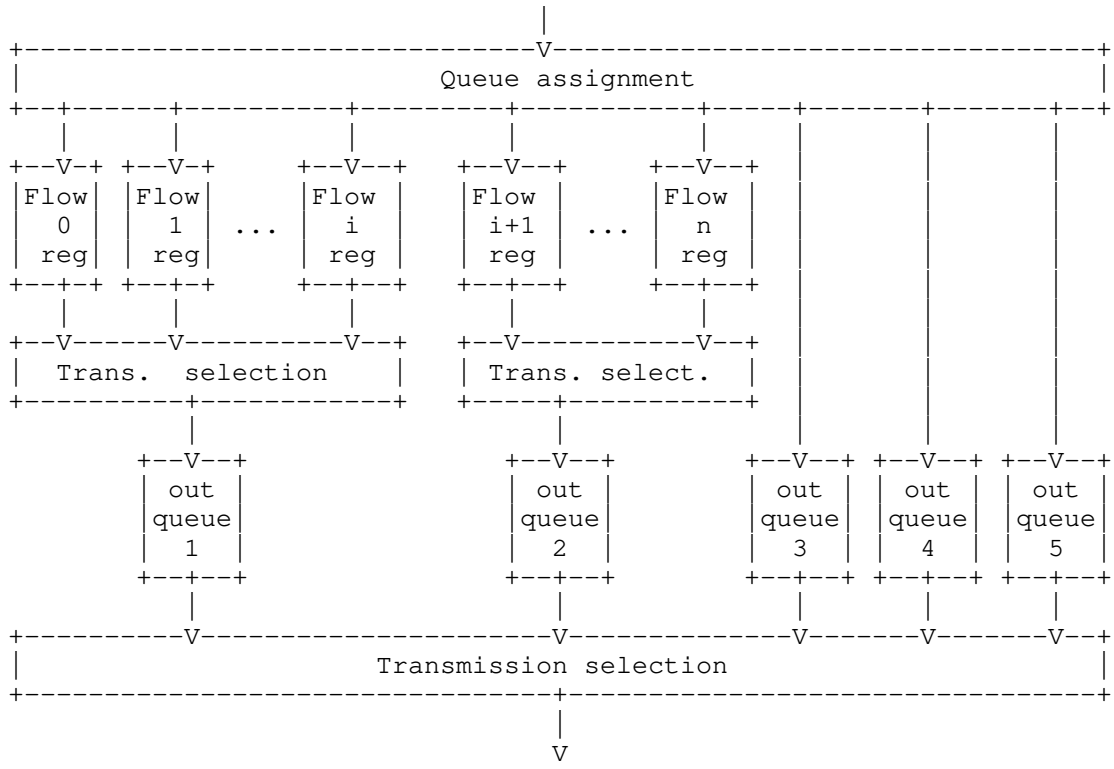


Figure 3: IEEE 802.1Q Queuing Model: Data flow

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

* Discarding packets because a queue is full.

* Discarding packets marked "yellow" by a metering function, in preference to discarding "green" packets [RFC2697].

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

The queue assignment function can be quite complex, even in a bridge [IEEE8021Q], since the introduction of per-stream filtering and policing ([IEEE8021Q] clause 8.6.5.1). In addition to the Layer 2 priority expressed in the 802.1Q VLAN tag, a DetNet transit node can utilize the information from the non-exhaustive list below to assign a packet to a particular queue:

* Input port.

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

* MAC addresses, VLAN ID, IP addresses, Layer 4 port numbers, DSCP [RFC8939], [RFC8964].

* The queue assignment function can contain metering and policing functions.

* MPLS and/or pseudo-wire labels [RFC6658].

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

6.2. Frame Preemption

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

6.3. Time-Aware Shaper

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

Note that scheduled traffic service relies on a synchronized network and coordinated GCL configuration. Synthesis of GCL on multiple nodes in network is a scheduling problem considering all DetNet flows traversing the network, which is a non-deterministic polynomial-time hard (NP-hard) problem [Sch8021Qbv]. Also, at this writing, scheduled traffic service supports no more than eight traffic queues, typically using up to seven priority queues and at least one best effort.

6.4. Credit-Based Shaper with Asynchronous Traffic Shaping

In this queuing model, it is assumed that the DetNet nodes are FIFO. We consider the four traffic classes (Definition 3.268 of [IEEE8021Q]): control-data traffic (CDT), class A, class B, and best effort (BE) in decreasing order of priority. Flows of classes A and B are DetNet flows that are less critical than CDT (such as studio audio and video traffic, as in IEEE 802.1BA Audio-Video-Bridging). This model is a subset of Time-Sensitive Networking as described next.

Based on the timing model described in Figure 1, contention occurs only at the output port of a DetNet transit node; therefore, the focus of the rest of this subsection is on the regulator and queuing subsystem in the output port of a DetNet transit node. The input flows are identified using the information in (Section 5.1 of [RFC8939]). Then they are aggregated into eight macro flows based on their service requirements; we refer to each macro flow as a class. The output port performs aggregate scheduling with eight queues (queuing subsystems): one for CDT, one for class A flows, one for class B flows, and five for BE traffic denoted as BE0-BE4. The queuing policy for each queuing subsystem is FIFO. In addition, each node output port also performs per-flow regulation for class A and B flows using an interleaved regulator (IR), called Asynchronous Traffic Shaper [IEEE8021Qcr]. Thus, at each output port of a node, there is one interleaved regulator per-input port and per-class; the interleaved regulator is mapped to the regulator depicted in Figure 1. The detailed picture of scheduling and regulation architecture at a node output port is given by Figure 4. The packets received at a node input port for a given class are enqueued in the respective interleaved regulator at the output port. Then, the packets from all the flows, including CDT and BE flows, are enqueued in queuing subsystem; there is no regulator for CDT and BE flows.

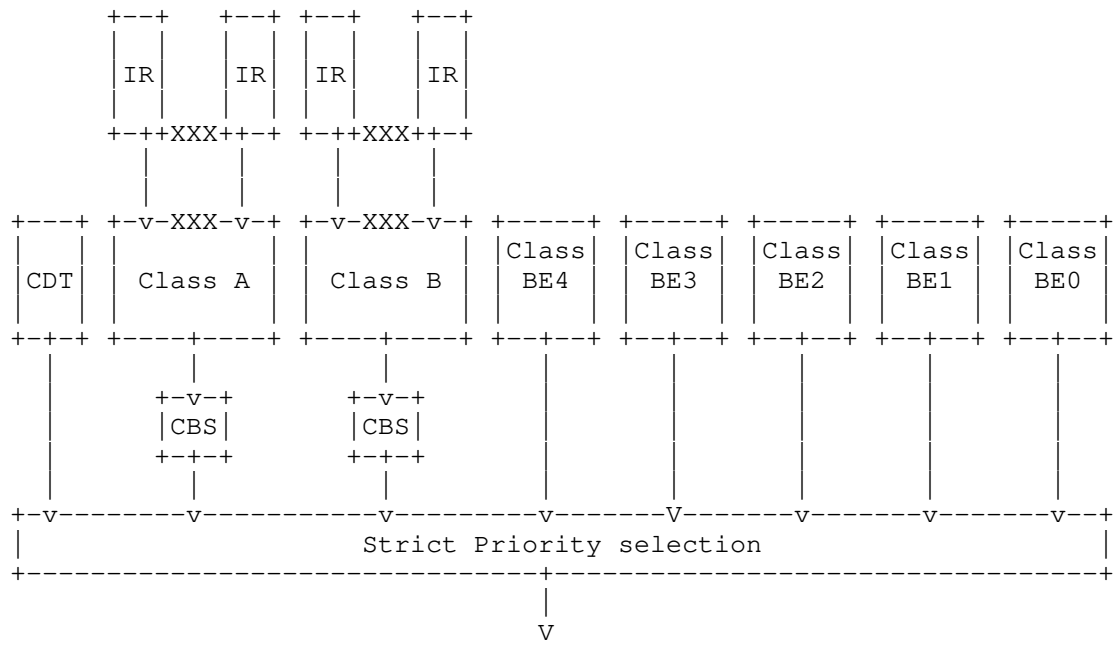


Figure 4: The architecture of an output port inside a relay node with interleaved regulators (IRs) and credit-based shaper (CBS)

Each of the queuing subsystems for classes A and B, contains a Credit-Based Shaper (CBS). The CBS serves a packet from a class according to the available credit for that class. As described in Section 8.6.8.2 and Annex L.1 of [IEEE8021Q], the credit for each class A or B increases based on the idle slope (as guaranteed rate), and decreases based on the sendslope (typically equal to the difference between the guaranteed and the output link rates), both of which are parameters of the CBS. The CDT and BE0-BE4 flows are served by separate queuing subsystems. Then, packets from all flows are served by a transmission selection subsystem that serves packets from each class based on its priority. All subsystems are non-preemptive. Guarantees for classes A and B traffic can be provided only if CDT traffic is bounded; it is assumed that the CDT traffic has a leaky bucket arrival curve with two parameters r_h as rate and b_h as bucket size, i.e., the amount of bits entering a node within a time interval t is bounded by r_h * t + b_h.

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

At each DetNet transit node implementing an interleaved regulator, packets of multiple flows are processed in one FIFO queue; the packet at the head of the queue is regulated based on its leaky bucket parameters; it is released at the earliest time at which this is possible without violating the constraint.

The regulation parameters for a flow (leaky bucket rate and bucket size) are the same at its source and at all DetNet transit nodes along its path in the case where all clocks are perfect. However, in reality there is clock non-ideality throughout the DetNet domain even with clock synchronization. This phenomenon causes inaccuracy in the rates configured at the regulators that may lead to network instability. To avoid that, when configuring the regulators, the rates are set as the source rates with some positive margin. [ThomasTime] describes and provides solutions to this issue.

6.4.1. Delay Bound Calculation

A delay bound of the queuing subsystem ((4) in Figure 1) of a given DetNet node for a flow of classes A or B can be computed if the following condition holds:

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

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

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

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

If the flow is of class A:

$$R_A = I_A * (c - r_h)/ c$$

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

where I_A is the idle slope for class A; L_nA is the maximum packet length of class B and BE packets; L_n is the maximum packet length of classes A,B, and BE; r_h is the rate and b_h is the bucket size of CDT traffic leaky bucket arrival curve.

If the flow is of class B:

$$R_B = I_B * (c-r_h)/ c$$

$$T_B = (L_{BE} + L_A + L_{nA} * I_A/(c_h-I_A) + b_h + r_h * L_n/ c)/(c-r_h)$$

where I_B is the idle slope for class B; L_A is the maximum packet length of class A; L_BE is the maximum packet length of class BE.

Then, as discussed in Section 4.2.2; an interleaved regulator does not increase the delay bound of the upstream queuing subsystem; therefore an end-to-end delay bound for a DetNet flow of class X (A or B) is the sum of d_X_i for all node i in the path the flow, where d_X_i is the delay bound of queuing subsystem in node i which is computed as above. According to the notation in Section 4.2.2, the delay bound of queuing subsystem in a node i and interleaved regulator in node j, i.e., Cij, is:

$$Cij = d_X_i$$

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

6.4.2. Flow Admission

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

The static admission control:

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

The dynamic admission control:

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

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

R_acc <=R, (Eq. 1)

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

b_acc <=b_t. (Eq. 2)

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

The choice of the static values of R and b_t at all nodes and classes must be done in a prior configuration phase; R controls the bandwidth allocated to this class at this node, b_t affects the delay bound and the buffer requirement. The value of R must be set such that

R <= I_X*(c-r_h)/c

where I_X is the idleslope of credit-based shaper for class X={A,B}, c is the transmission rate of the output link and r_h is the leaky-bucket rate of the CDT class.

6.5. Guaranteed-Service IntServ

Guaranteed-Service Integrated service (IntServ) is an architecture that specifies the elements to guarantee quality of service (QoS) on networks [RFC2212].

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

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

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

The provided delay bound is based on a simple case of Guaranteed-Service IntServ where only a guaranteed rate and maximum service latency and a leaky bucket arrival curve are available. If more information about the flow is known, e.g., the peak rate, the delay bound is more complicated; the details are available in [RFC2212] and Section 1.4.1 of [NetCalBook].

6.6. Cyclic Queuing and Forwarding

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

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

The cycle time T_c must be carefully chosen; it needs to be large enough to accommodate all the DetNet traffic, plus at least one maximum packet (or fragment) size from lower priority queues, which might be received within a cycle. Also, the value of T_c includes a time interval, called dead time (DT), which is the sum of the delays 1,2,3,4 defined in Figure 1. The value of DT guarantees that the

last packet of one cycle in a node is fully delivered to a buffer of the next node in the same cycle. A two-buffer CQF is recommended if DT is small compared to T_c. For a large DT, CQF with more buffers can be used, and a cycle identification label can be added to the packets.

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

(h+1) T_c

and the minimum latency is:

(h-1) T_c + DT.

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

7. Example application on DetNet IP network

This section provides an example application of the timing model presented in this document to control the admission of a DetNet flow on a DetNet-enabled IP network. Consider Figure 5, taken from Section 3 of [RFC8939], that shows a simple IP network:

* The end-system 1 implements Guaranteed-Service IntServ as in Section 6.5 between itself and relay node 1.

* Sub-network 1 is a TSN network. The nodes in subnetwork 1 implement credit-based shapers with asynchronous traffic shaping as in Section 6.4.

* Sub-network 2 is a TSN network. The nodes in subnetwork 2 implement cyclic queuing and forwarding with two buffers as in Section 6.6.

* The relay nodes 1 and 2 implement credit-based shapers with asynchronous traffic shaping as in Section 6.4. They also perform the aggregation and mapping of IP DetNet flows to TSN streams (Section 4.4 of [RFC9023]).

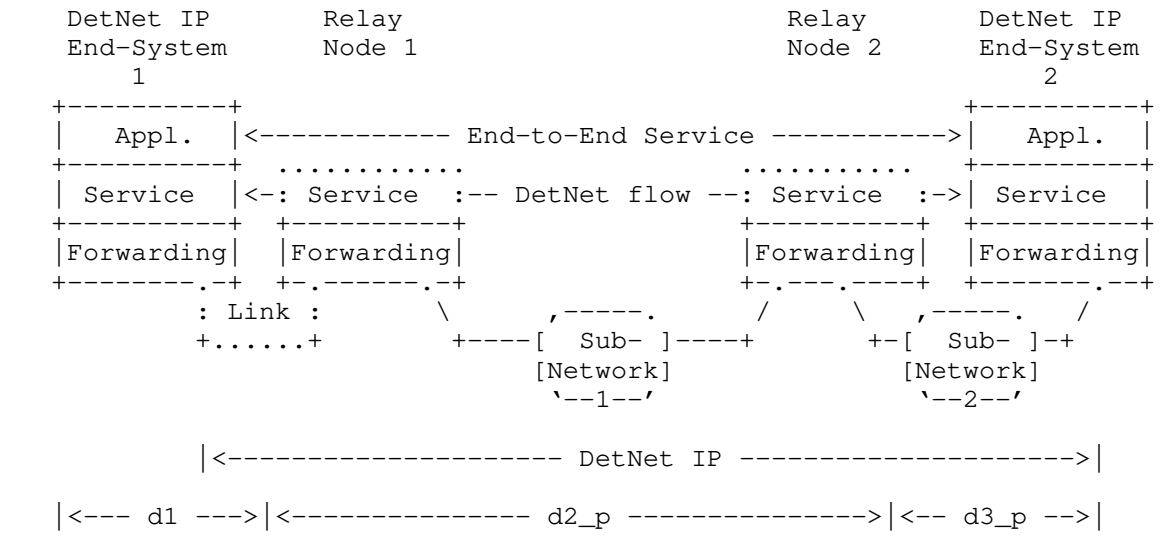


Figure 5: A Simple DetNet-Enabled IP Network, taken from RFC8939

Consider a fully centralized control plane for the network of Figure 5 as described in Section 3.2 of [I-D.ietf-detnet-controller-plane-framework]. Suppose end-system 1 wants to create a DetNet flow with traffic specification destined to end-system 2 with end-to-end delay bound requirement D. Therefore, the control plane receives a flow establishment request and calculates a number of valid paths through the network (Section 3.2 of [I-D.ietf-detnet-controller-plane-framework]). To select a proper path, the control plane needs to compute an end-to-end delay bound at every node of each selected path p.

The end-to-end delay bound is d1 + d2_p + d3_p, where d1 is the delay bound from end-system 1 to the entrance of relay node 1, d2_p is the delay bound for path p from relay node 1 to entrance of the first node in sub-network 2, and d3_p the delay bound of path p from the first node in sub-network 2 to end-system 2. The computation of d1 is explained in Section 6.5. Since the relay node 1, sub-network 1 and relay node 2 implement aggregate queuing, we use the results in Section 4.2.2 and Section 6.4 to compute d2_p for the path p. Finally, d3_p is computed using the delay bound computation of Section 6.6. Any path p such that d1 + d2_p + d3_p <= D satisfies the delay bound requirement of the flow. If there is no such path, the control plane may compute new set of valid paths and redo the delay bound computation or reject the DetNet flow.

As soon as the control plane selects a path that satisfies the delay bound constraint, it allocates and reserves the resources in the path for the DetNet flow (Section 4.2 [I-D.ietf-detnet-controller-plane-framework]).

8. Security considerations

Detailed security considerations for DetNet are cataloged in [RFC9055], and more general security considerations are described in [RFC8655].

Security aspects that are unique to DetNet are those whose aim is to provide the specific QoS aspects of DetNet, specifically bounded end-to-end delivery latency and zero congestion loss. Achieving such loss rates and bounded latency may not be possible in the face of a highly capable adversary, such as the one envisioned by the Internet Threat Model of BCP 72 [RFC3552] that can arbitrarily drop or delay any or all traffic. In order to present meaningful security considerations, we consider a somewhat weaker attacker who does not control the physical links of the DetNet domain but may have the ability to control or change the behavior of some resources within the boundary of the DetNet domain.

Latency bound calculations use parameters that reflect physical quantities. If an attacker finds a way to change the physical quantities, unknown to the control and management planes, the latency calculations fail and may result in latency violation and/or congestion losses. An example of such attacks is to make some traffic sources under the control of the attacker send more traffic than their assumed T-SPECs. This type of attack is typically avoided by ingress conditioning at the edge of a DetNet domain. However, it must be insured that such ingress conditioning is done per-flow and that the buffers are segregated such that if one flow exceeds its T-SPEC, it does not cause buffer overflow for other flows.

Some queuing mechanisms require time synchronization and operate correctly only if the time synchronization works correctly. In the case of CQF, the correct alignments of cycles can fail if an attack against time synchronization fools a node into having an incorrect offset. Some of these attacks can be prevented by cryptographic authentication as in Annex K of [IEEE1588] for the Precision Time Protocol (PTP). However, the attacks that change the physical latency of the links used by the time synchronization protocol are still possible even if the time synchronization protocol is protected by authentication and cryptography [DelayAttack]. Such attacks can be detected only by their effects on latency bound violations and congestion losses, which do not occur in normal DetNet operation.

9. IANA considerations

This document has no IANA actions.

10. Acknowledgement

We would like to thank Lou Berger, Tony Przygienda, John Scudder, Watson Ladd, Yoshifumi Nishida, Ralf Weber, Robert Sparks, Gyan Mishra, Martin Duke, Eric Vyncke, Lars Eggert, Roman Danyliw, and Paul Wouters for their useful feedback on this document.

11. Contributors

RFC 7322 limits the number of authors listed on the front page to a maximum of 5. The editor wishes to thank and acknowledge the following author for contributing text to this document

Janos Farkas
Ericsson
Email: janos.farkas@ericsson.com

12. References

12.1. Normative References

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

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

[RFC2475] Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z., and W. Weiss, "An Architecture for Differentiated Services", RFC 2475, DOI 10.17487/RFC2475, December 1998, <https://www.rfc-editor.org/info/rfc2475>.

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

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

[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>.

[RFC8964] Varga, B., Ed., Farkas, J., Berger, L., Malis, A., Bryant, S., and J. Korhonen, "Deterministic Networking (DetNet) Data Plane: MPLS", RFC 8964, DOI 10.17487/RFC8964, January 2021, <https://www.rfc-editor.org/info/rfc8964>.

[RFC9016] Varga, B., Farkas, J., Cummings, R., Jiang, Y., and D. Fedyk, "Flow and Service Information Model for Deterministic Networking (DetNet)", RFC 9016, DOI 10.17487/RFC9016, March 2021, <https://www.rfc-editor.org/info/rfc9016>.

12.2. Informative References

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

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

[DelayAttack]
S. Barreto, A. Suresh, and J.-Y. Le Boudec, "Cyber-attack on packet-based time synchronization protocols: The undetectable Delay Box", <https://ieeexplore.ieee.org/document/7520408>.

[I-D.ietf-detnet-controller-plane-framework]
A. Malis, X. Geng, M. Chen, F. Qin, and B. Varga, "Deterministic Networking (DetNet) Controller Plane Framework draft-ietf-detnet-controller-plane-framework-01", <https://datatracker.ietf.org/doc/html/draft-ietf-detnet-controller-plane-framework>.

[IEEE1588] IEEE Std 1588-2008, "IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems", 2008, <https://ieeexplore.ieee.org/document/4579760>.

[IEEE8021Qcr]
IEEE 802.1, "IEEE P802.1Qcr: Bridges and Bridged Networks - Amendment: Asynchronous Traffic Shaping", 2017, <https://1.ieee802.org/tsn/802-1qcr/>.

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

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

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

[NetCalBook]
J.-Y. Le Boudec and P. Thiran, "Network calculus: a theory of deterministic queuing systems for the internet", 2001, <https://leboudec.github.io/netcal/>.

[PacketReorderingBounds]
E. Mohammadpour, and J.-Y. Le Boudec, "On Packet Reordering in Time-Sensitive Networks", <https://ieeexplore.ieee.org/document/9640523>.

[RFC2697] Heinanen, J. and R. Guerin, "A Single Rate Three Color Marker", RFC 2697, DOI 10.17487/RFC2697, September 1999, <https://www.rfc-editor.org/info/rfc2697>.

[RFC3552] Rescorla, E. and B. Korver, "Guidelines for Writing RFC Text on Security Considerations", BCP 72, RFC 3552, DOI 10.17487/RFC3552, July 2003, <https://www.rfc-editor.org/info/rfc3552>.

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

[RFC9023] Varga, B., Ed., Farkas, J., Malis, A., and S. Bryant, "Deterministic Networking (DetNet) Data Plane: IP over IEEE 802.1 Time-Sensitive Networking (TSN)", RFC 9023, DOI 10.17487/RFC9023, June 2021, <https://www.rfc-editor.org/info/rfc9023>.

[RFC9055] Grossman, E., Ed., Mizrahi, T., and A. Hacker, "Deterministic Networking (DetNet) Security Considerations", RFC 9055, DOI 10.17487/RFC9055, June 2021, <https://www.rfc-editor.org/info/rfc9055>.

[Sch8021Qbv] S. Craciunas, R. Oliver, M. Chmelik, and W. Steiner, "Scheduling Real-Time Communication in IEEE 802.1Qbv Time Sensitive Networks", <https://dl.acm.org/doi/10.1145/2997465.2997470>.

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

[ThomasTime] L. Thomas and J.-Y. Le Boudec, "On Time Synchronization Issues in Time-Sensitive Networks with Regulators and Nonideal Clocks", <https://dl.acm.org/doi/10.1145/3393691.3394206>.

[TSNwithATS] E. Mohammadpour, E. Stai, M. Mohiuddin, and J.-Y. Le Boudec, "Latency and Backlog Bounds in Time-Sensitive Networking with Credit Based Shapers and Asynchronous Traffic Shaping", <https://ieeexplore.ieee.org/document/8493026>.

Authors' Addresses

Norman Finn
Huawei Technologies Co. Ltd
3101 Rio Way
Spring Valley, California 91977
United States of America
Phone: +1 925 980 6430
Email: nfinn@nfinnconsulting.com

Jean-Yves Le Boudec
EPFL
IC Station 14
CH-1015 Lausanne EPFL
Switzerland
Email: jean-yves.leboudec@epfl.ch

Ehsan Mohammadpour
EPFL
IC Station 14
CH-1015 Lausanne EPFL
Switzerland
Email: ehsan.mohammadpour@epfl.ch

Jiayi Zhang
Huawei Technologies Co. Ltd
Q27, No.156 Beiqing Road
Beijing
100095
China
Email: zhangjiayi11@huawei.com

Balázs Varga
Ericsson
Budapest
Konyves Kálmán krt. 11/B
1097
Hungary
Email: balazs.a.varga@ericsson.com

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

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

October 12, 2020

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

Abstract

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

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

Requirements Language

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

Status of This Memo

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

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

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

This Internet-Draft will expire on April 15, 2021.

Copyright Notice

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

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

Table of Contents

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

1. Introduction

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

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

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

2. Terminologies

This documents uses the terminologies defined in [RFC8655].

3. DetNet Configuration Module

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

3.1. DetNet Appliction Flow Configuration Attributes

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

3.2. DetNet Service Sub-layer Configuration Attributes

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

- DetNet flow identification
- Service function indication, indicates which service function will be invoked at a DetNet edge, relay node or end station. (DetNet tunnel initialization or termination are default functions in DetNet service layer, so there is no need for explicit indication). The corresponding arguments for service functions also needs to be defined.

3.3. DetNet Forwarding Sub-layer Configuration Attributes

As defined in [RFC8655], DetNet forwarding sub-layer optionally provides congestion protection for DetNet flows over paths provided by the underlying network. Explicit route is another mechanism that

is used by DetNet to avoid temporary interruptions caused by the convergence of routing or bridging protocols, and it is also implemented at the DetNet forwarding sub-layer.

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

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

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

4. DetNet Flow Aggregation

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

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

- Aggregation flow encapsulation/decapsulation/identification

- Mapping individual DetNet flows to an aggregated flow

- Changing traffic specification parameters for aggregated flow

The following cases of DetNet aggregation are supported:

- aggregate data flows into an application which is then mapped to a service sub-layer at the ingress node. Note the data flows may be other DetNet flows.

- map each DetNet application to a single service sub-layer and allowing the aggregation of multiple applications at the ingress

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

- o map each DetNet application uniquely to a single service sub-layer where those sub-layers may be encapsulated as a single service sub-layer and hence aggregating the applications at the ingress node, and vice versa for de-aggregation. In this case, the service sub-layer identifier may be sufficient to identify the application. A classifier may be required to de-aggregate the service sub-layers.

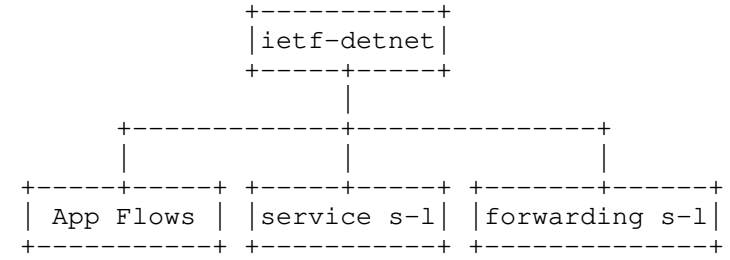
- o aggregate DetNet service sub-layers into an aggregated flow by using the same forwarding sub-layer at ingress node or relay node, and vice versa for de-aggregation.

- o aggregate DetNet flows with different forwarding sub-layer into an aggregated flow by using the same forwarding sub-layer at transit node, and vice versa for de-aggregation.

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

5. DetNet YANG Structure Considerations

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



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

6. DetNet Configuration YANG Structures

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

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

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

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

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

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

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

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

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

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

7. DetNet Configuration YANG Model

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

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

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

  import ietf-ethertypes {
    prefix "eth";
  }

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

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

  organization
    "IETF DetNet Working Group";

  contact
```

```
    "WG Web:   <http://tools.ietf.org/wg/detnet/>
     WG List:  <mailto: detnet@ietf.org>
     WG Chair: Lou Berger
                 <mailto:lberger@labn.net>

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```
  typedef service-protection-type {
    type enumeration {
      enum none {
        description
          "no service protection provide";
      }
      enum replication {
        description
          "A Packet Replication Function (PRF) replicates
           DetNet flow packets and forwards them to one or
           more next hops in the DetNet domain.  The number
           of packet copies sent to each next hop is a
           DetNet flow specific parameter at the node doing
           the replication.  PRF can be implemented by an
           edge node, a relay node, or an end system";
      }
      enum elimination {
        description
          "A Packet Elimination Function (PEF) eliminates
           duplicate copies of packets to prevent excess
           packets flooding the network or duplicate
           packets being sent out of the DetNet domain.
           PEF can be implemented by an edge node, a relay
           node, or an end system.";
      }
      enum ordering {
        description
          "A Packet Ordering Function (POF) re-orders
           packets within a DetNet flow that are received
           out of order.  This function can be implemented
           by an edge node, a relay node, or an end system.";
      }
      enum elimination-ordering {
        description
          "A combination of PEF and POF that can be
           implemented by an edge node, a relay node, or
           an end system.";
      }
      enum elimination-replication {
        description
          "A combination of PEF and PRF that can be
           implemented by an edge node, a relay node, or
           an end system";
      }
      enum elimination-ordering-replicaiton {
        description
          "A combination of PEF, POF and PRF that can be
           implemented by an edge node, a relay node, or
```

```
           an end system";
      }
    }
  }

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

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

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

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

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

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

    leaf next-header {
      type uint8;
      description
        "The next header of the IPv6 header";
    }

    leaf traffic-class {
      type uint8;
      description
        "The traffic class value of the header";
    }

    leaf flow-label {
      type inet:ipv6-flow-label;
      description
        "The flow label value of the header";
    }

    leaf source-port {
      type inet:port-number;
      description
        "The source port number";
    }

    leaf destination-port {
      type inet:port-number;
      description
        "The destination port number";
    }
  }

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

```
    }

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

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

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

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

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

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

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

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

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

    leaf next-header {
      type uint8;
      description
        "The next header of the IPv6 header";
    }

    leaf traffic-class {
      type uint8;
      description
        "The traffic class value of the header";
    }

    leaf flow-label {
      type inet:ipv6-flow-label;
      description
        "The flow label value of the header";
    }

    uses source-ip-port-identification;

    uses destination-ip-port-identification;

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

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

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

  grouping traffic-specification {
    container traffic-specification {
      description
        "traffic-specification specifies how the Source
         transmits packets for the flow.  This is the
         promise/request of the Source to the network.
         The network uses this traffic specification
         to allocate resources and adjust queue
         parameters in network nodes.";
      reference
        "draft-ietf-detnet-flow-information-model";
      leaf interval {
        type uint32;
        description
          "The period of time in which the traffic
           specification cannot be exceeded";
      }

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

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

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

      leaf average-payload-size {
```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

          case mpls {
```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```
      uses traffic-requirements;

      uses traffic-specification;

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

        uses aggregation-header;

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

      container incoming {
        uses aggregation-header;

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

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

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

        leaf service-rank {
          type uint8;
          description
            "The DetNet rank for this service";
        }

        choice service-type {
          mandatory true;
          container non-grouped {
            leaf traffic-profile {
              type traffic-profile-ref;
              description
                "The Traffic Profile for this service";
            }

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

          container grouped {
            leaf group-ref {
              type aggregation-grp-ref;
              description
                "The aggregation group this service belongs to";
            }
          }
        }

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

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

```
            filed";
        }
      }

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

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

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

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

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

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

```
                uses detnet-header;

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

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

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

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

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

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

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

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

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

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

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

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

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

              uses detnet-flow-spec;
            }

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

```
        }

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

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

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

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

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

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

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

8. Open Issues

There are some open issues that are still under discussion:

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

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

9. IANA Considerations

This document makes no request of IANA.

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

10. Security Considerations

<TBD>

11. Acknowledgements

12. References

12.1. Normative References

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

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <https://www.rfc-editor.org/info/rfc6991>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

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

12.2. Informative References

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

Authors' Addresses

Xuesong Geng
Huawei Technologies

Email: gengxuesong@huawei.com

Mach(Guoyi) Chen
Huawei Technologies

Email: mach.chen@huawei.com

Yeoncheol Ryoo
ETRI

Email: dbduscjf@etri.re.kr

Don Fedyk
LabN Consulting, L.L.C.

Email: dfedyk@labn.net

Reshad Rahman
Cisco Systems

Email: rrahman@cisco.com

Zhenqiang Li
China Mobile

Email: lizhenqiang@chinamobile.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: 26 August 2024

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

23 February 2024

Deterministic Networking (DetNet) YANG Model
draft-ietf-detnet-yang-20

Abstract

This document contains the specification for the Deterministic Networking YANG Model for configuration and operational data of DetNet Flows. The model allows for provisioning of end-to-end DetNet service on devices 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).

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 26 August 2024.

Copyright Notice

Copyright (c) 2024 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction 3
2. Abbreviations 3
3. Terminology 3
4. DetNet YANG Module 4
4.1. DetNet Application Flow YANG Attributes 5
4.2. DetNet Service Sub-layer YANG Attributes 5
4.3. DetNet Forwarding Sub-layer YANG Attributes 5
5. DetNet Flow Aggregation 6
6. DetNet YANG Structure Considerations 7
7. DetNet Configuration YANG Structures 8
8. DetNet Configuration YANG Model 10
9. IANA Considerations 40
10. Security Considerations 41
11. Contributors 42
12. Acknowledgments 42
13. References 42
13.1. Normative References 42
13.2. Informative References 44
Appendix A. DetNet Configuration YANG Tree 45
Appendix B. Examples 55
B.1. Example A-1 JSON Configuration/Operational 55
B.2. Example B-1 XML Config: Aggregation using a Forwarding Sub-layer 60
B.3. Example B-2 JSON Service Aggregation Configuration . . . 64
B.4. Example C-1 JSON Relay Aggregation/Disaggregation Configuration 70
B.5. Example C-2 JSON Relay Aggregation/Disaggregation Service Sub-Layer 87
B.6. Example C-3 JSON Relay Service Sub-Layer Aggregation/ Disaggregation 99
B.7. Example C-4 JSON Relay Service Sub-Layer Aggregation/ Disaggregation 113
B.8. Example D-1 JSON Transit Forwarding Sub-Layer Aggregation/ Disaggregation 129
Authors' Addresses 136

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.

This Yang model is scoped to the description of the aggregation/ disaggregation and data plane capabilities of the DetNet data planes defined in the DetNet Architecture [RFC8655] and DetNet Framework [RFC8938]. DetNet operates at the IP layer and delivers service over lower-layer technologies such as MPLS and IEEE 802.1 Time-Sensitive Networking (TSN).

2. Abbreviations

The following abbreviations are used in this document:

PEF	Packet Elimination Function
PRF	Packet Replication Function
PEOF	Packet Elimination and Ordering Functions
PERF	Packet Elimination and Replication Functions
PREOF	Packet Replication, Elimination and Ordering Functions
MPLS	Multiprotocol Label Switching

3. Terminology

This document uses the terminology defined in [RFC8655]. The terms A-label, S-label, and F-label are used in this document as defined in [RFC8964].

4. 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 4.1 , Section 4.2 , Section 4.3 respectively.

Layers of the objects typically occur in the different data instances forming the node types defined in [RFC8655]. Figure 1 illustrates the relationship between data instance node types and the included layers. Node types are logical roles per DetNet service: a device along one DetNet service can be of one node type, while another service may use the same device with a different node type. This model is a controller based model because a controller or operator configures all the devices to form a service.

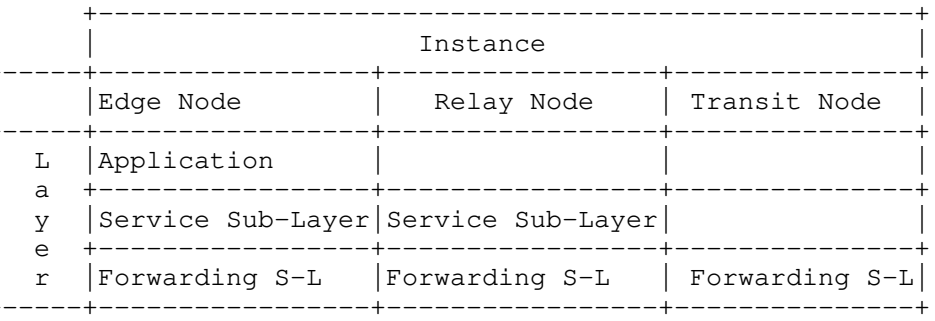


Figure 1: DetNet Layers and Node Types

All of the layers have ingress/incoming and egress/outgoing operations, but any instance may be configured as only unidirectional. Ingress refers to any DetNet layer where a DetNet context is applied. Ingress allows functions such as switching, aggregation and encapsulation. Likewise, egress refers to any DetNet layer where a DetNet context is removed. Egress allows functions such as switching, disaggregation and decapsulation. This means that each unidirectional flow identifier configuration is programmed starting at the ingress and flow status is reported at ingress on each end. In the MPLS cases once encapsulated, the IP 6-tuple, see [RFC8938], parameters may not be required to be programmed again. In the IP case, without encapsulation, various IP flow id parameters must be configured along the flow path.

In the YANG model the terms source and destination are used as flow identifiers whereas ingress and egress refer to a DetNet application direction from the application edge. Ingress is to the DetNet

application and egress is from the application. The terms incoming and outgoing generally represent the flow direction towards the remote application. Outgoing is viewed as going down the stack from Application to Service sub-layer to Forwarding sub-layer and incoming is the reverse. Although, in examples where there is aggregation and disaggregation outgoing relates to the aggregating output and incoming relates to the disaggregating flows.

At the egress point, forwarding information is determined by the App-flow type with all DetNet-related headers removed. The forwarding information can specify an output port, or set a next-hop-address in case of IP, or set an MPLS label in case of MPLS.

4.1. 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 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 [RFC9016].

4.2. 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 the DetNet service layer, so there is no need for explicit indication). The corresponding arguments for service functions also need to be defined.

4.3. 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, as described in the information model in [RFC9016]. These may be 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 needs 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 or other aspects.

5. DetNet Flow Aggregation

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

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

* Aggregated flow encapsulation/decapsulation/identification

* Mapping individual DetNet flows to an aggregated flow

* Changing traffic specification parameters for aggregated flows

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

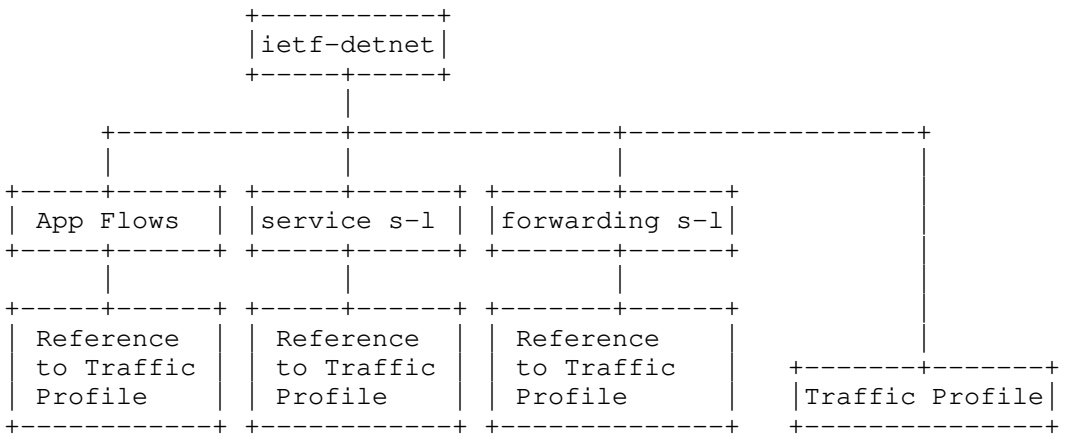
* Relay node aggregates the forwarding sub-layers of DetNet flow into a service sub-layer of Aggregated DetNet flow

* 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.

6. DetNet YANG Structure Considerations

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



There are three layer types in the DetNet YANG Model: App-flow data layer, service sub-layer and forwarding sub-layer. Additionally, the Traffic parameters are captured in a Traffic profile that can be referenced by any of the layers.

Below is a summary YANG tree showing the major items. A complete YANG tree is in section Appendix A.

A traffic profile can be created for an application, a service sub-layer or a forwarding sub-layer. A single profile may be shared by multiple applications/sub-layer. Each profile indicates the members currently using that profile.

Depending on which DetNet layers and functions are required, some or all of the components may be configured. Examples are shown in Appendix B.

7. DetNet Configuration YANG Structures

The following is a partial tree representation of the YANG as defined in [RFC8340]. This corresponds to the structure layout in the previous section.

```
module: ietf-detnet
  +--rw detnet
     +--rw traffic-profile* [name]
     |  +--rw name                      string
     |  +--rw traffic-requirements
     |  +--rw traffic-spec
     |  +--ro member-app-flow*             app-flow-ref
     |  +--ro member-svc-sublayer*         service-sub-layer-ref
     |  +--ro member-fwd-sublayer*   forwarding-sub-layer-ref
     +--rw app-flows
     |  +--rw app-flow* [name]
     |     +--rw name                 string
     |     +--rw 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 egress
     |           ...
     +--rw service
     |  +--rw sub-layer* [name]
     |     +--rw name                   string
     |     +--rw service-rank?          uint8
     |     +--rw traffic-profile?       traffic-profile-ref
     |     +--rw service-protection
     |     |     ...
     |     +--rw operation?             operation
     |     +--rw incoming
     |     |     ...
     |     +--rw outgoing
     |           ...
     +--rw forwarding
        +--rw sub-layer* [name]
           +--rw name                string
           +--rw traffic-profile?    traffic-profile-ref
           +--rw operation?          mpls-fwd-operation
           +--rw incoming
           |     ...
           +--rw outgoing
                 ...
```

8. DetNet Configuration YANG Model

This YANG model imports typedefs from [RFC6991], [RFC8519], [RFC8294], [RFC8343], and [IEEE8021Q]. This YANG model also has the following references to RFCs that are not in the document text body [RFC0791], [RFC4303], [RFC8349], [RFC8938], [RFC8960], [RFC8964], and [RFC8200].

```
<CODE BEGINS> file "ietf-detnet@2022-02-21.yang"
module ietf-detnet {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-detnet";
  prefix dnet;

  import ietf-yang-types {
    prefix yang;
    reference
      "RFC 6991 - 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 802.1Q-2022 - IEEE Standard for Local and Metropolitan
         Area Networks--Bridges and Bridged Networks Clause 48: YANG
         Data Models.";
    }

    organization
      "IETF DetNet Working Group";

    contact
      "WG Web:   <https://datatracker.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:   Zhenqiang Li
                 <mailto:lizhenqiang@chinamobile.com>";
    description
      "This YANG module describes the parameters needed
       for DetNet flow configuration and flow status
       reporting. This YANG module conforms to the Network
       Management Datastore Architecture (NMDA).

       Copyright (c) 2024 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 Revised 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;
       see the RFC itself for full legal notices.";

       // RFC Ed.: replace XXXX with actual RFC number and remove
       // this note
```

```
        // replace '2024-02-21' with the module publication date
        // the format is (year-month-day)

      revision 2024-02-21 {
        description
          "Initial revision";
        reference
          "RFC XXXX: Deterministic Networking (DetNet) YANG Model";
      }

      identity app-status {
        description
          "Base identity from which all application-status
           status types are derived.";
        reference
          "RFC 9016 Section 5.8";
      }

      identity none {
        base app-status;
        description
          "This application has no status. This identity is
           expected when the configuration is incomplete.";
        reference
          "RFC 9016 Section 5.8";
      }

      identity ready {
        base app-status;
        description
          "Application ingress/egress ready.";
        reference
          "RFC 9016 Section 5.8";
      }

      identity failed {
        base app-status;
        description
          "Application ingres/egress failed.";
        reference
          "RFC 9016 Section 5.8";
      }

      identity out-of-service {
        base app-status;
        description
          "Application administratively blocked.";
        reference
```

```
        "RFC 9016 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
        "RFC 9016 Section 5.8";
    }

    typedef app-flow-ref {
      type leafref {
        path "/dnet:detnet"
           + "/dnet:app-flows"
           + "/dnet:app-flow"
           + "/dnet:name";
      }
      description
        "This is an application Reference.";
    }

    typedef service-sub-layer-ref {
      type leafref {
        path "/dnet:detnet"
           + "/dnet:service"
           + "/dnet:sub-layer"
           + "/dnet:name";
      }
      description
        "This is a service sub-layer Reference.";
    }

    typedef forwarding-sub-layer-ref {
      type leafref {
        path "/dnet:detnet"
           + "/dnet:forwarding"
           + "/dnet:sub-layer"
           + "/dnet:name";
      }
      description
        "This is a forwarding sub-layer Reference.";
    }

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

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

    typedef ipsec-spi {
      type uint32 {
        range "1..max";
      }
      description
        "IPsec Security Parameters Index. A 32 bit value
         where some values are reserved.";
      reference
        "IETF RFC 4303 Encapsulating Security Payload (ESP).";
    }

    typedef operation {
      type enumeration {
        enum initiation {
          description
            "This is an initiating service sub-layer encapsulation.";
        }
        enum termination {
          description
            "Operation for DetNet service sub-layer decapsulation.";
        }
        enum 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. Operations are described as unidirectional
         but a service sub-layer may combine operation types.";
    }

    typedef mpls-fwd-operation {
      type enumeration {
        enum impose-and-forward {
          description
```

```
            "This operation imposes outgoing label(s) and forwards to
             next-hop.";
          reference
            " A YANG Data Model for MPLS Base RFC 8960.";
        }
        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 RFC 8960.";
        }
        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 RFC 8960.";
        }
        enum swap-and-forward {
          description
            "This operation swaps an incoming label, with an outgoing
             label and forwards to the next-hop.";
          reference
            " A YANG Data Model for MPLS Base RFC 8960.";
        }
        enum forward {
          description
            "This operation forwards to next-hop.";
        }
        enum pop-and-lookup {
          description
            "This operation pops an incoming label and performs a
             lookup.";
        }
      }
      description
        "MPLS operations types.  This is an enum modeled after the
         MPLS enum.  The enums are the same as A YANG Data Model
         for MPLS Base. RFC 8960.";
    }

    typedef service-protection {
      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-replication {
          description
            "A combination of PEF, POF and PRF that can be implemented
             by an edge node, a relay node, or an end system.";
        }
      }
      description
        "This typedef describes the service protection enumeration
         values.";
    }

    typedef sequence-number-generation {
      type enumeration {
        enum copy-from-app-flow {
```

```
          description
            "Copy-from-app-flow is used to extend and use the
            sequence number used in App-flow. This function is
            required when encapsulating App-flows that have been
            replicated and received through multiple ingress nodes
            into a member flow, and then eliminate it at the relay
            node.";
        }
        enum generate-by-detnet-flow {
          description
            "Generate-by-detnet-flow is used to create a new
            sequence number for a DetNet flow at the ingress node.
            Care must be taken when using this option to ensure
            there is only one source for generating sequence
            numbers.";
        }
      }
      description
        "This typedef defines how to generate sequence numbers to
        be used in DetNet encapsulation.";
    }
    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 enumeration configures 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-no-zone;
```

```
        description
          "The source IP address in the header.";
        reference
          "RFC 6991 Common YANG Data Types";
      }
      leaf dest-ip-address {
        type inet:ip-address-no-zone;
        description
          "The destination IP address in the header.";
        reference
          "RFC 6991 Common YANG Data Types";
      }
      leaf protocol-next-header {
        type uint8;
        description
          "In IPv4 refers to the protocol of the
           payload.  In IPv6, this field is known as 'next-header',
           and identifies the type of header immediately following
           the IPv6 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 6991 Common YANG Data Types";
      }
      leaf flow-label {
        type inet:ipv6-flow-label;
        description
          "The flow label value of the header. IPv6 only.";
        reference
          "RFC 6991 Common YANG Data Types";
      }
      leaf source-port {
        type inet:port-number;
        description
          "The source port number.";
        reference
          "RFC 6991 Common YANG Data Types";
      }
      leaf destination-port {
        type inet:port-number;
        description
```

```
          "The destination port number.";
        reference
          "RFC 6991 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 802.1Q-2022.";
      }
      leaf pcp {
        type dot1q-types:priority-type;
        description
          "The priority value of the Ethernet header.";
        reference
          "IEEE 802.1Q-2022.";
      }
    }

    grouping destination-ip-port-id {
      description
        "The TCP/UDP port 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 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 6991 Common YANG Data Types";
      }
      leaf dest-ip-prefix {
        type inet:ip-prefix;
        description
          "The destination IP prefix.";
        reference
          "RFC 6991 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 6991 Common YANG Data Types";
      }
      leaf flow-label {
        type inet:ipv6-flow-label;
        description
          "The flow label value of the header.";
        reference
          "RFC 6991 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 4303 Encapsulating Security Payload (ESP).";
      }
    }

    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 flow {
        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 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 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 mpls {
            description
              "MPLS label stack for DetNet MPLS encapsulation or
               forwarding.";
            uses rt-types:mpls-label-stack;
          }
          case ip {
            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. This follows the
           principles for next hops in RFC 8349";
        choice next-hop-options {
          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.";
          leaf outgoing-interface {
            type if:interface-ref;
            description
              "The outgoing interface, when matching all flows to
               the 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.";
            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
                "The outgoing interface, when matching all flows to
                 the 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. This follows the
         principles for next hops in RFC 8349";
      choice next-hop-options {
        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.";
          leaf outgoing-interface {
            type if:interface-ref;
            description
              "The outgoing interface, when matching all flows to
               the interface.";
          }
          choice flow-type {
            description
              "These are the flow type next hop choices.";
            case ip {
              description
                "Use IP data plane for forwarding.";
              leaf next-hop-address {
                type inet:ip-address;
                description
                  "This is an IP address as a next hop.";
              }
              uses ip-header;
```

```
            }
            case mpls {
              description
                "Use MPLS data plane for forwarding.";
              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.";
            leaf hop-index {
              type uint8;
              description
                "The value of the index for a hop.";
            }
            leaf outgoing-interface {
              type if:interface-ref;
              description
                "The outgoing interface, when matching all flows to
                 the interface.";
            }
            choice flow-type {
              description
                "These are the flow type next hop choices.";
              case ip {
                description
                  "Use IP data plane for forwarding.";
                leaf next-hop-address {
                  type inet:ip-address;
                  description
                    "This is an IP address as a next hop.";
                }
                uses ip-header;
              }
              case mpls {
                description
                  "Use MPLS data plane for forwarding.";
                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 "name";
        description
          "A traffic profile.";
        leaf name {
          type string;
          description
            "The name of the traffic profile which is used as a
             reference to this profile.";
        }
        container traffic-requirements {
          description
            "This defines the attributes of the App-flow
             regarding bandwidth, latency, latency variation, loss, and
             misordering tolerance.";
          reference
            "RFC 9016 Section 5.9";
          leaf min-bandwidth {
            type uint64;
            units 'octets per second';
            description
              "This is the minimum bandwidth that has to be
               guaranteed for the DetNet service.  MinBandwidth is
               specified in octets per second.";
            reference
              "RFC 9016 Section 5.9.1";
          }
          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. The MAX value is 4,294,967,295
               nanoseconds.";
            reference
              "RFC 9016 Section 5.9.2";
          }
          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.";
            reference
              "RFC 9016 Section 5.9.3";
          }
          leaf max-loss {
            type decimal64 {
              fraction-digits 10;
              range "0 .. 100";
            }
            units "percent";
            description
              "This defines the maximum Packet Loss Rate (PLR)
              parameter for the DetNet service between the Ingress and
              Egress(es) of the DetNet domain. Packet loss rate is
              calculated by the number of transmitted packets minus
              the number of received packets divided by the number
              transmitted packets expressed as a percent.";
            reference
              "RFC 9016 Section 5.9.4";
          }
          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.";
            reference
              "RFC 9016 Section 5.9.5";
          }
          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.";
```

```
            reference
              "RFC 9016 Section 5.9.6";
          }
        }
        container traffic-spec {
          description
            "Traffic-specification specifies how the Source transmits
             packets for the flow.  This is the promise/request of the
             Source to the network.  The network uses this flow
             specification to allocate resources and adjust queue
             parameters in network nodes.";
          reference
            "RFC 9016 Section 5.5";
          leaf interval {
            type uint32;
            units "nanoseconds";
            description
              "The period of time in which the traffic
               specification should not be exceeded.";
            reference
              "RFC 9016 Section 5.5,
               IEEE802.1Q";
          }
          leaf max-pkts-per-interval {
            type uint32;
            description
              "The maximum number of packets that the
               source will transmit in one interval.";
            reference
              "RFC 9016 Section 5.5, IEEE802.1Q";
          }
          leaf max-payload-size {
            type uint32;
            description
              "The maximum payload size that the source
               will transmit.";
            reference
              "RFC 9016 Section 5.5, IEEE802.1Q";
          }
          leaf min-payload-size {
            type uint32;
            description
              "The minimum payload size that the source
               will transmit., IEEE802.1Q";
          }
          leaf min-pkts-per-interval {
            type uint32;
            description
```

```
            "The minimum number of packets that the
             source will transmit in one interval.";
          reference
            "RFC 9016 Section 5.5, IEEE802.1Q";
        }
      }
      leaf-list member-app-flow {
        type app-flow-ref;
        config false;
        description
          "A list of Applications attached to this profile.  Each
          application that uses a profile has an automatically
          populated reference.";
        reference
          "RFC XXXX: Deterministic Networking (DetNet) YANG Model
          Section 5";
      }
      leaf-list member-svc-sublayer {
        type service-sub-layer-ref;
        config false;
        description
          "A list of Service Sub-layers attached to this profile.
          Each Service Sub-layers that uses a profile has an
          automatically populated reference.";
        reference
          "RFC XXXX: Deterministic Networking (DetNet) YANG Model
          Section 5";
      }
      leaf-list member-fwd-sublayer {
        type forwarding-sub-layer-ref;
        config false;
        description
          "A list of Forwarding Sub-layers attached to this profile.
          Each Forwarding Sub-layers that uses a profile has an
          automatically populated reference.";
        reference
          "RFC XXXX: Deterministic Networking (DetNet) YANG Model
          Section 5";
      }
    }
    container app-flows {
      description
        "The DetNet app-flow configuration.";
      reference
        "RFC 9016 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
              "RFC 9016
               Sections 4.1, 5.1";
          }
          leaf 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
              "RFC 9016
               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 app-flow-status {
              type identityref {
                base app-status;
              }
              default none;
              config false;
```

```
            description
              "Status of ingress application flow. This is an
               operational status and defaults to none if
               incomplete.";
            reference
              "RFC 9016 Sections
               4.1, 5.8";
          }
          leaf-list interface {
            type if:interface-ref;
            description
            "Interface is optional for a service type. When
             matching a flow to a single interface one
             interface is specified. The list allows for
             matching a subset of interfaces. When more
             than one interfaces is specified, these
             flows are simply aggregated and the service
             sub-layer is unaware of the aggregation.";
          }
          uses data-flow-spec;
        } //End of app-ingress
        container egress {
          description
            "Egress DetNet application flows or a compound flow.";
          uses data-flow-spec;
          choice application-type {
            description
              "This is the application type choices.";
            container ethernet {
              description
                "This is Ethernet or TSN traffic that maps to an
                interface.";
              leaf-list interface {
                type if:interface-ref;
                description
                  "This is one or more Ethernet or TSN interfaces.
                   If multiple interfaces are specified, this
                   application flow is replicated to those
                   interfaces.  DetNet application Flow filtering
                   applies to the whole list of interfaces.
                   For fine grain flow filtering, use a single
                   interface per application.";
              }
            }
            container ip-mpls {
              description
                "This is IP or MPLS DetNet application types.";
              uses detnet-app-next-hop-content;
```

```
            }
          }
        }
      }
    }
    container service {
      description
        "The DetNet service sub-layer configuration.";
      list sub-layer {
        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;
          default 255;
          description
            "The DetNet rank for this service. Defaults to 255
            lowest rank if not specified.";
          reference
            "RFC 9016 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 and sequence number
             options.";
          leaf protection {
            type service-protection;
            description
              "The DetNet service protection type such as
              Packet Replication Function (PRF),
              Packet Elimination Function (PEF),
              Packet Replication, Elimination, and Ordering Functions
              (PREOF).";
            reference
              "RFC 8938 Section 4.3";
          }
          leaf sequence-number-length {
            type sequence-number-field;
```

```
            default zero-sn;
            description
              "Sequence number field length can be one of 0 (none),
               16-bits or 28-bits. The default is none.";
          }
        }
        leaf operation {
          type operation;
          description
            "This is the service operation type for this service
             sub-layer;";
        }
        container incoming {
          description
            "The DetNet service sub-layer incoming configuration.";
          choice incoming {
            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 forwarding-sub-layer {
            description
              "This entry specifies one or more forwarding
               sub-layers. No or minimal service sub-layer
               encapsulation is allowed.";
            leaf-list sub-layer {
              type forwarding-sub-layer-ref;
              config false;
              description
                "List of outgoing forwarding sub-layers.";
             }
          }
        }
      }
      container outgoing {
        description
          "The DetNet service sub-layer outgoing configuration.";
        choice outgoing {
          description
            "The outgoing type may be a forwarding Sub-layer or a
             service sub-layer or aggregation type.";
          container forwarding-sub-layer {
            description
              "This service sub-layer is sending 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.";
            reference
              "RFC 8964 Section 4.2.1 and 4.2.2.";
            list service-outgoing {
              key "index";
              description
                "List of the outgoing service
                 that separately for each node
                 where services will be eliminated.";
              leaf 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 sending 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.";
              reference
                "RFC 8964 Section 4.2.1 and 4.2.2.";
              leaf aggregation-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. This
                  is optional and only used when the service
                  sub-layer uses MPLS. It is an MPLS stack since
                  more than a single label may be used.";
                uses rt-types:mpls-label-stack;
              }
            }
            container app-flow {
              description
                "This service sub-layer is sending 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.";
              reference
                "RFC 8964 Section 4.2.1 and 4.2.2.";
              uses app-flows-group;
            }
            container service-disaggregation {
              description
                "This service sub-layer is sending 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.";
                  reference
                    "RFC 8964 Section 4.2.1 and 4.2.2.";
                  uses service-sub-layer-group;
                }
                container forwarding-disaggregation {
                  description
                    "This service sub-layer is sending 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.";
                  reference
                    "RFC 8964 Section 4.2.1 and 4.2.2.";
                  uses forwarding-sub-layer-group;
                }
              }
            }
          }
        }
        container forwarding {
          description
            "The DetNet forwarding sub-layer configuration.";
          list sub-layer {
            key "name";
            description
              "The list is one or more DetNet service/forwarding 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 operation {
              type mpls-fwd-operation;
              description
                "This is the forwarding operation types
                 impose-and-forward, pop-and-forward,
```

```
            pop-impose-and-forward, forward, pop-and-lookup.";
        }
        container incoming {
          description
            "The DetNet forwarding sub-layer incoming
             configuration.";
          choice incoming {
            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 {
          description
            "The DetNet forwarding sub-layer outbound
             configuration.";
          choice outgoing {
            description
              "This is when a service connected directly to an
```

```
          interface with no forwarding sub-layer.";
        container
          interface {
          description
            "This forwarding sub-layer is sending 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 sending to the service
             sub-layers of the lower layer for
             forwarding-to-service aggregation at the ingress
             node or relay node.";
          leaf aggregation-sub-layer {
            type service-sub-layer-ref;
            description
              "This is a 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 sending 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-sub-layer {
            type forwarding-sub-layer-ref;
            description
              "This is a 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 sending 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;
                  reference
                    "RFC 8964 Section 4.2.3";
                }
                container forwarding-disaggregation {
                  description
                    "This forwarding sub-layer is sending 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>
```

9. IANA Considerations

This document registers a URI in the "IETF XML Registry" [RFC3688]. Following the format in [RFC3688], the following registration is requested to be made:

```
ID:  yang:ietf-detnet
URI:  urn:ietf:params:xml:ns:yang:ietf-detnet
Registrant Contact:  The IESG.
XML:  N/A, the requested URI is an XML namespace.
```

This document registers YANG modules in the "YANG Module Names" registry [RFC6020].

```
Name:  ietf-detnet
Maintained by IANA:  N
Namespace:  urn:ietf:params:xml:ns:yang:ietf-detnet
Prefix:  dnet
Reference:  This RFC when published.
```

10. Security Considerations

Security considerations for DetNet are covered in the DetNet Architecture [RFC8655] and DetNet Security Considerations [RFC9055] .

The YANG modules specified in this document define a schema for data that is designed to be accessed via network management protocols, such as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC8446].

The Network Configuration Access Control Model (NACM) [RFC8341] provides the means to restrict access for particular NETCONF or RESTCONF users to a preconfigured subset of all available NETCONF or RESTCONF protocol operations and content.

There are a number of data nodes defined in the module that are writable/creatable/deletable (i.e., config true, which is the default). These data nodes may be considered sensitive or vulnerable in some network environments. Unauthorized write operations (e.g., edit-config) to any elements of this module can break or incorrectly connect DetNet flows. Since DetNet is a configured Data Plane, any changes that are not coordinated with all devices along the path will create a denial of service. In addition, arbitrary write operations could also enable an attacker to modify a network path to enable select traffic to avoid inspection or treatment by security controls, or route traffic in a way that it would be subject to inspect/ modification by an adversary node.

Similarly, the data nodes in these YANG modules may be considered sensitive or vulnerable in some network environments. It is thus important to control read access (e.g., via get, get-config, or notification) to these data nodes. These are the subtrees and data node and their sensitivity/vulnerability:

/detnet/app-flows: This controls the application details so it could be considered sensitive.

/detnet/traffic-profile/member-app-flow: This links traffic profiles to applications, service sub-layers and/or and forwarding sub-layers so this also could be considered more sensitive.

/detnet/service/sub-layer/incoming/app-flow: This links applications to services.

/detnet/service/sub-layer/outgoing/app-flow: This links applications to services.

The above nodes can reveal identifiable characteristics of the application flows.

/detnet/service/sub-layer: This defines the service and forwarding operations.

/detnet/fowarding/sub-layer: This defines the forwarding operations.

The above nodes can reveal some aspects of the network topology of there is unauthorized access to this configuration.

11. Contributors

The editors of this document wish to thank and acknowledge the following people who contributed substantially to the content of this document and should be considered coauthors:

Mach(Guoyi) Chen
Huawei Technologies
Email: mach.chen@huawei.com

12. Acknowledgments

The editors of this document would like to thank Lou Berger, Tom Petch Xufeng Lui, Julien Meuric, John Scudder‘ and Florian Kauer for their detailed comments.

13. References

13.1. Normative References

[RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, DOI 10.17487/RFC0791, September 1981, <https://www.rfc-editor.org/info/rfc791>.

[RFC4303] Kent, S., "IP Encapsulating Security Payload (ESP)", RFC 4303, DOI 10.17487/RFC4303, December 2005, <https://www.rfc-editor.org/info/rfc4303>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <https://www.rfc-editor.org/info/rfc6020>.

[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>.

[RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <https://www.rfc-editor.org/info/rfc8200>.

[RFC8294] Liu, X., Qu, Y., Lindem, A., Hopps, C., and L. Berger, "Common YANG Data Types for the Routing Area", RFC 8294, DOI 10.17487/RFC8294, December 2017, <https://www.rfc-editor.org/info/rfc8294>.

[RFC8343] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 8343, DOI 10.17487/RFC8343, March 2018, <https://www.rfc-editor.org/info/rfc8343>.

[RFC8349] Lhotka, L., Lindem, A., and Y. Qu, "A YANG Data Model for Routing Management (NMDA Version)", RFC 8349, DOI 10.17487/RFC8349, March 2018, <https://www.rfc-editor.org/info/rfc8349>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

[RFC8519] Jethanandani, M., Agarwal, S., Huang, L., and D. Blair, "YANG Data Model for Network Access Control Lists (ACLs)", RFC 8519, DOI 10.17487/RFC8519, March 2019, <https://www.rfc-editor.org/info/rfc8519>.

[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>.

[RFC8938] Varga, B., Ed., Farkas, J., Berger, L., Malis, A., and S. Bryant, "Deterministic Networking (DetNet) Data Plane Framework", RFC 8938, DOI 10.17487/RFC8938, November 2020, <https://www.rfc-editor.org/info/rfc8938>.

[RFC8960] Saad, T., Raza, K., Gandhi, R., Liu, X., and V. Beeram, "A YANG Data Model for MPLS Base", RFC 8960, DOI 10.17487/RFC8960, December 2020, <https://www.rfc-editor.org/info/rfc8960>.

[RFC8964] Varga, B., Ed., Farkas, J., Berger, L., Malis, A., Bryant, S., and J. Korhonen, "Deterministic Networking (DetNet) Data Plane: MPLS", RFC 8964, DOI 10.17487/RFC8964, January 2021, <https://www.rfc-editor.org/info/rfc8964>.

[RFC9016] Varga, B., Farkas, J., Cummings, R., Jiang, Y., and D. Fedyk, "Flow and Service Information Model for Deterministic Networking (DetNet)", RFC 9016, DOI 10.17487/RFC9016, March 2021, <https://www.rfc-editor.org/info/rfc9016>.

13.2. Informative References

[IEEE8021Q] IEEE, "IEEE Standard for Local and Metropolitan Area Networks--Bridges and Bridged Networks", DOI 10.1109/IEEESTD.2022.10004498, IEEE 802.1Q-2022, July 2022, <https://ieeexplore.ieee.org/document/8403927>.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://www.rfc-editor.org/info/rfc3688>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/info/rfc6241>.

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <https://www.rfc-editor.org/info/rfc6242>.

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <https://www.rfc-editor.org/info/rfc8040>.

[RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <https://www.rfc-editor.org/info/rfc8340>.

[RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <https://www.rfc-editor.org/info/rfc8341>.

[RFC9055] Grossman, E., Ed., Mizrahi, T., and A. Hacker, "Deterministic Networking (DetNet) Security Considerations", RFC 9055, DOI 10.17487/RFC9055, June 2021, <https://www.rfc-editor.org/info/rfc9055>.

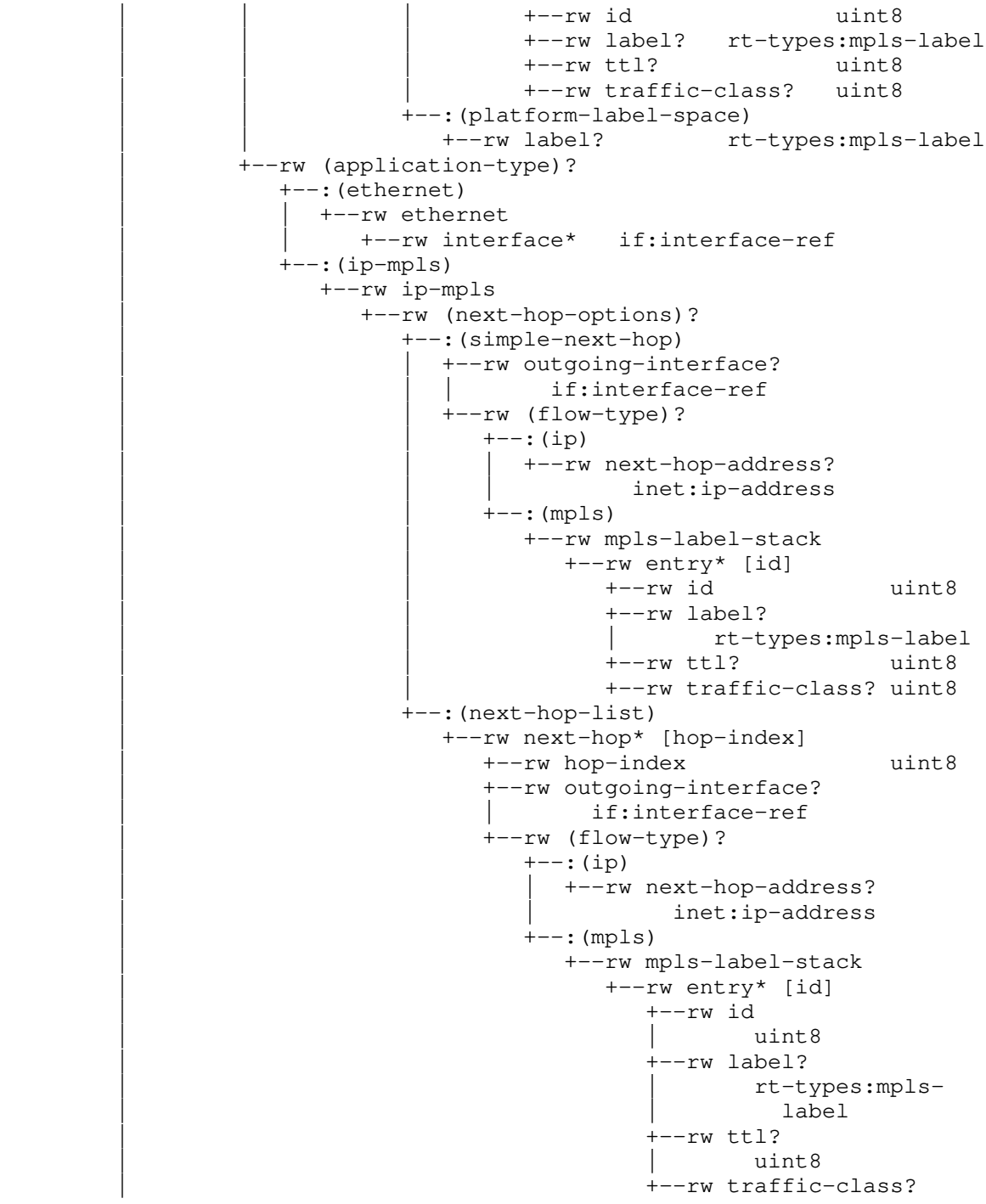
Appendix A. DetNet Configuration YANG Tree

This is the full YANG tree as described in [RFC8340].

```
module: ietf-detnet
  +--rw detnet
     +--rw traffic-profile* [name]
     |  +--rw name                       string
     |  +--rw traffic-requirements
     |  |  +--rw min-bandwidth?                    uint64
     |  |  +--rw max-latency?                      uint32
     |  |  +--rw max-latency-variation?            uint32
     |  |  +--rw max-loss?                         decimal64
     |  |  +--rw max-consecutive-loss-tolerance?   uint32
     |  |  +--rw max-misordering?                  uint32
     |  +--rw traffic-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-app-flow*            app-flow-ref
     |  +--ro member-svc-sublayer*        service-sub-layer-ref
     |  +--ro member-fwd-sublayer*   forwarding-sub-layer-ref
     +--rw app-flows
     |  +--rw app-flow* [name]
     |     +--rw name                string
     |     +--rw 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
     |     |  +--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 (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 (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
        |  +--rw sub-layer* [name]
        |     +--rw name                    string
        |     +--rw service-rank?           uint8
        |     +--rw traffic-profile?        traffic-profile-ref
        |     +--rw service-protection
        |     |  +--rw protection?                service-protection
        |     |  +--rw sequence-number-length?    sequence-number-field
        |     +--rw operation?              operation
        |     +--rw incoming
        |     |  +--rw (incoming)?
        |     |     +--:(app-flow)
        |     |     |  +--rw app-flow
        |     |     |     +--rw flow*   app-flow-ref
        |     |     +--:(service-aggregation)
        |     |     |  +--rw service-aggregation
        |     |     |     +--rw sub-layer*   service-sub-layer-ref
        |     |     +--:(forwarding-aggregation)
        |     |     |  +--rw forwarding-aggregation
        |     |     |     +--rw 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
        |     |     +--:(forwarding-sub-layer)
        |     |        +--rw forwarding-sub-layer
        |     |           +--ro sub-layer*   forwarding-sub-layer-ref
        |     +--rw outgoing
        |        +--rw (outgoing)?
        |           +--:(forwarding-sub-layer)
        |           |  +--rw forwarding-sub-layer
        |           |     +--rw service-outgoing* [index]
        |           |        +--rw index                         uint8
        |           |        +--rw (header-type)?
        |           |        |  +--:(mpls)
        |           |        |  |  +--rw mpls-label-stack
        |           |        |  |     +--rw entry* [id]
        |           |        |  |        +--rw id               uint8
        |           |        |  |        +--rw label?
        |           |        |  |        |       rt-types:mpls-label
        |           |        |  |        +--rw ttl?             uint8
        |           |        |  |        +--rw traffic-class?   uint8
        |           |        |  +--:(ip)
        |           |        |     +--rw src-ip-address?
        |           |        |     |       inet:ip-address-no-zone
        |           |        |     +--rw dest-ip-address?
        |           |        |     |       inet:ip-address-no-zone
        |           |        |     +--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 sub-layer*
|            |                   forwarding-sub-layer-ref
|            +--:(service-sub-layer)
|            |  +--rw service-sub-layer
|            |     +--rw aggregation-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 flow*   app-flow-ref
|            +--:(service-disaggregation)
|            |  +--rw service-disaggregation
|            |     +--rw sub-layer*   service-sub-layer-ref
|            +--:(forwarding-disaggregation)
|               +--rw forwarding-disaggregation
|                  +--rw sub-layer*   forwarding-sub-layer-ref
|
+--rw forwarding
   +--rw sub-layer* [name]
      +--rw name                string
      +--rw traffic-profile?    traffic-profile-ref
      +--rw operation?          mpls-fwd-operation
      +--rw incoming
      |  +--rw (incoming)?
      |     +--:(service-sub-layer)
      |     |  +--rw service-sub-layer
      |     |     +--rw sub-layer*   service-sub-layer-ref
      |     +--:(forwarding-aggregation)
      |     |  +--rw forwarding-aggregation
      |     |     +--rw 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
   +--rw (outgoing)?
      +--:(interface)
      |  +--rw interface
      |     +--rw (next-hop-options)?
```

```
|        +--:(simple-next-hop)
|        |  +--rw outgoing-interface?
|        |  |       if:interface-ref
|        |  +--rw (flow-type)?
|        |     +--:(ip)
|        |     |  +--rw next-hop-address?
|        |     |  |       inet:ip-address
|        |     |  +--rw src-ip-address?
|        |     |  |       inet:ip-address-no-zone
|        |     |  +--rw dest-ip-address?
|        |     |  |       inet:ip-address-no-zone
|        |     |  +--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 next-hop-address?
|                 |  |       inet:ip-address
|                 |  +--rw src-ip-address?
|                 |  |       inet:ip-address-no-zone
|                 |  +--rw dest-ip-address?
|                 |  |       inet:ip-address-no-zone
|                 |  +--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-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-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 sub-layer*   service-sub-layer-ref
|                 +--:(forwarding-disaggregation)
|                    +--rw forwarding-disaggregation
|                       +--rw sub-layer*   forwarding-sub-layer-ref
```

Appendix B. 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. Note that IPv4 and IPv6 addresses are supported but for clarity in the examples and diagrams IPv4 has been used in most examples. The IP types are imported from [RFC6991] and these support both IPv4 and IPv6.

The following conventions are used in the diagrams.

* Replication and Elimination points are shown as an R in and E in circles respectively.

* Packet Headers including DetNet aggregation label or A-label, Service label or S-label and Forwarding label or F-label are illustrated at each hop as defined in [RFC8964].

* Aggregation/Disaggregation nodes are indicated by dashed line boxes.

* Since the model augments IETF interfaces, minimal interface YANG data is provided to validate the interface data as well. This shows up as a named value such as "eth0" etc. that is referenced by the configuration.

The following are examples of aggregation and disaggregation at various points in DetNet. Figures are provided in the PDF and HTML version of this document.

B.1. Example A-1 JSON Configuration/Operational

This illustrates that multiple App flows with the same source, the same destination, and the same traffic specification are aggregated in a single DetNet flow service sub-layer. Ingress node 1 aggregates App flows 0 and 1 into a service sub-layer of DetNet flow 1. Two ways of illustrating this follow, then the JSON operational data model corresponding to the diagrams follows. This example uses IPv6 address format.

Please consult the PDF or HTML versions for the Case A-1 Diagram.

Figure 2: Case A-1 Application Aggregation

Please consult the PDF or HTML versions for the Case A-1 Diagram.

Figure 3: Case A-1 Application Aggregation Flow Stack Detail

Figure 4 contains the operational JSON configuration for the ingress aggregation node illustrated in Figure 2 and Figure 3. App-0 and app-1 are aggregated into Service Sub-layer ssl-1.

```
{
  "ietf-detnet:detnet": {
    "traffic-profile": [
      {
        "name": "pf-1",
        "traffic-requirements": {
          "min-bandwidth": "100000000",
          "max-latency": 100000000,
          "max-latency-variation": 20000000,
          "max-loss": "0.0000001",
          "max-consecutive-loss-tolerance": 5,
          "max-misordering": 0
        },
        "traffic-spec": {
          "interval": 5,
          "max-pkts-per-interval": 10,
          "max-payload-size": 1500,
          "min-payload-size": 100,
          "min-pkts-per-interval": 1
        },
        "member-app-flow": [
          "app-0",
          "app-1"
        ]
      },
      {
        "name": "pf-2",
        "traffic-requirements": {
          "min-bandwidth": "200000000",
          "max-latency": 100000000,
          "max-latency-variation": 20000000,
          "max-loss": "0.000001",
          "max-consecutive-loss-tolerance": 5,
          "max-misordering": 0
        },
        "traffic-spec": {
          "interval": 5,
          "max-pkts-per-interval": 20,
          "max-payload-size": 1500,
          "min-payload-size": 100,
          "min-pkts-per-interval": 1
        },
        "member-svc-sublayer": [
          "ssl-1"
```

```
          ]
        },
        {
          "name": "pf-3",
          "traffic-spec": {
            "interval": 5,
            "max-pkts-per-interval": 10,
            "max-payload-size": 1500
          },
          "member-fwd-sublayers": [
            "fsl-1"
          ]
        }
      ],
      "app-flows": {
        "app-flow": [
          {
            "name": "app-0",
            "bidir-congruent": false,
            "outgoing-service": "ssl-1",
            "traffic-profile": "pf-1",
            "ingress": {
              "app-flow-status": "ietf-detnet:ready",
              "interface": [
                "eth0"
              ],
              "ip-app-flow": {
                "src-ip-prefix": "2001:db8::1/128",
                "dest-ip-prefix": "2001:db8::8/128",
                "dscp": 6
              }
            }
          },
          {
            "name": "app-1",
            "bidir-congruent": false,
            "outgoing-service": "ssl-1",
            "traffic-profile": "pf-1",
            "ingress": {
              "app-flow-status": "ietf-detnet:ready",
              "interface": [
                "eth0"
              ],
              "ip-app-flow": {
                "src-ip-prefix": "2001:db8::1/128",
                "dest-ip-prefix": "2001:db8::8/128",
                "dscp": 7
              }
```

```
            }
          }
        ]
      },
      "service": {
        "sub-layer": [
          {
            "name": "ssl-1",
            "service-rank": 10,
            "traffic-profile": "pf-2",
            "service-protection": {
              "protection": "none",
              "sequence-number-length": "long-sn"
            },
            "operation": "initiation",
            "incoming": {
              "app-flow": {
                "flow": [
                  "app-0",
                  "app-1"
                ]
              }
            },
            "outgoing": {
              "forwarding-sub-layer": {
                "service-outgoing": [
                  {
                    "index": 0,
                    "mpls-label-stack": {
                      "entry": [
                        {
                          "id": 0,
                          "label": 100
                        }
                      ]
                    },
                    "sub-layer": [
                      "fsl-1"
                    ]
                  }
                ]
              }
            }
          }
        ]
      },
      "forwarding": {
        "sub-layer": [
```

```
          {
            "name": "fsl-1",
            "traffic-profile": "pf-3",
            "operation": "impose-and-forward",
            "incoming": {
              "service-sub-layer": {
                "sub-layer": [
                  "ssl-1"
                ]
              }
            },
            "outgoing": {
              "interface": {
                "outgoing-interface": "eth2",
                "mpls-label-stack": {
                  "entry": [
                    {
                      "id": 0,
                      "label": 10000
                    }
                  ]
                }
              }
            }
          }
        ]
      }
    },
    "ietf-interfaces:interfaces": {
      "interface": [
        {
          "name": "eth0",
          "type": "iana-if-type:ethernetCsmacd",
          "oper-status": "up",
          "statistics": {
            "discontinuity-time": "2024-02-21T18:59:00-05:00"
          }
        },
        {
          "name": "eth2",
          "type": "iana-if-type:ethernetCsmacd",
          "oper-status": "up",
          "statistics": {
            "discontinuity-time": "2024-02-21T18:59:00-05:00"
          }
        }
      ]
    }
```

```
}
```

Figure 4: Example A-1 DetNet configuration Application Aggregation

B.2. Example B-1 XML Config: Aggregation using a Forwarding Sub-layer

Figure 5 illustrates the DetNet service sub-layer flows 1 and 2 are aggregated into a single forwarding sub-layer. For the same destination multiple DetNet flows use a single forwarding path and service protection is performed by the corresponding service sub-layer of each flow. A diagram illustrating this case is shown and then the corresponding XML operational data for node Ingress 1 follows.

Please consult the PDF or HTML versions for the Case B-1 Diagram.

Figure 5: Case B-1 Example Config: Aggregation using a Forwarding Sub-layer

Figure 6 contains the operational XML configuration for the ingress aggregation node illustrated in Figure 5. In this example app-0 and app-1 are in separate service sub-layers with MPLS labels and the aggregation happens at the forwarding sub-layer afl-1 using MPLS labels.

```
<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>2024-02-21T23:59:00Z</discontinuity-time>
      </statistics>
    </interface>
    <interface>
      <name>eth1</name>
      <type>ia:ethernetCsmacd</type>
      <oper-status>up</oper-status>
      <statistics>
        <discontinuity-time>2024-02-21T23:59:00Z</discontinuity-time>
      </statistics>
    </interface>
    <interface>
      <name>eth2</name>
      <type>ia:ethernetCsmacd</type>
      <oper-status>up</oper-status>
```

```
        <statistics>
          <discontinuity-time>2024-02-21T23:59:00Z</discontinuity-time>
        </statistics>
      </interface>
    </interfaces>
  <detnet
    xmlns="urn:ietf:params:xml:ns:yang:ietf-detnet">
    <app-flows>
      <app-flow>
        <name>app-1</name>
        <bidir-congruent>false</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>192.0.2.1/32</src-ip-prefix>
            <dest-ip-prefix>192.0.2.8/32</dest-ip-prefix>
            <dscp>6</dscp>
          </ip-app-flow>
        </ingress>
      </app-flow>
      <app-flow>
        <name>app-2</name>
        <bidir-congruent>false</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>192.0.2.2/32</src-ip-prefix>
            <dest-ip-prefix>192.0.2.9/32</dest-ip-prefix>
            <dscp>7</dscp>
          </ip-app-flow>
        </ingress>
      </app-flow>
    </app-flows>
    <traffic-profile>
      <name>1</name>
      <traffic-requirements>
        <min-bandwidth>100000000</min-bandwidth>
        <max-latency>100000000</max-latency>
        <max-latency-variation>20000000</max-latency-variation>
        <max-loss>0.0000001</max-loss>
        <max-consecutive-loss-tolerance>5
          </max-consecutive-loss-tolerance>
```

```
        <max-misordering>0</max-misordering>
      </traffic-requirements>
      <traffic-spec>
        <interval>5</interval>
        <max-pkts-per-interval>10</max-pkts-per-interval>
        <max-payload-size>1500</max-payload-size>
      </traffic-spec>
      <member-app-flow>app-1</member-app-flow>
      <member-app-flow>app-2</member-app-flow>
    </traffic-profile>
    <traffic-profile>
      <name>2</name>
      <traffic-requirements>
        <min-bandwidth>100000000</min-bandwidth>
        <max-latency>100000000</max-latency>
        <max-latency-variation>20000000</max-latency-variation>
        <max-loss>0.000001</max-loss>
        <max-consecutive-loss-tolerance>5
          </max-consecutive-loss-tolerance>
        <max-misordering>0</max-misordering>
      </traffic-requirements>
      <member-svc-sublayer>ssl-1</member-svc-sublayer>
      <member-svc-sublayer>ssl-2</member-svc-sublayer>
    </traffic-profile>
    <traffic-profile>
      <name>3</name>
      <traffic-spec>
        <interval>5</interval>
        <max-pkts-per-interval>20</max-pkts-per-interval>
        <max-payload-size>1500</max-payload-size>
      </traffic-spec>
      <member-fwd-sublayer>afl-1</member-fwd-sublayer>
    </traffic-profile>
    <service>
      <sub-layer>
        <name>ssl-1</name>
        <service-rank>10</service-rank>
        <traffic-profile>2</traffic-profile>
        <operation>initiation</operation>
        <service-protection>
          <protection>none</protection>
          <sequence-number-length>long-sn</sequence-number-length>
        </service-protection>
       <incoming>
          <app-flow>
            <flow>app-1</flow>
          </app-flow>
        </incoming>
```

```
          <outgoing>
            <forwarding-sub-layer>
              <service-outgoing>
                <index>0</index>
                <mpls-label-stack>
                  <entry>
                    <id>0</id>
                    <label>100</label>
                  </entry>
                </mpls-label-stack>
                <sub-layer>afl-1</sub-layer>
              </service-outgoing>
             </forwarding-sub-layer>
          </outgoing>
        </sub-layer>
        <sub-layer>
          <name>ssl-2</name>
          <service-rank>10</service-rank>
          <traffic-profile>2</traffic-profile>
          <operation>initiation</operation>
          <service-protection>
            <protection>none</protection>
            <sequence-number-length>long-sn</sequence-number-length>
          </service-protection>
         <incoming>
            <app-flow>
              <flow>app-2</flow>
            </app-flow>
          </incoming>
          <outgoing>
            <forwarding-sub-layer>
              <service-outgoing>
                <index>0</index>
                <mpls-label-stack>
                  <entry>
                    <id>0</id>
                    <label>103</label>
                  </entry>
                </mpls-label-stack>
                <sub-layer>afl-1</sub-layer>
              </service-outgoing>
             </forwarding-sub-layer>
          </outgoing>
        </sub-layer>
        </service>
        <forwarding>
        <sub-layer>
          <name>afl-1</name>
```

```
        <traffic-profile>3</traffic-profile>
        <operation>impose-and-forward</operation>
        <incoming>
          <service-sub-layer>
            <sub-layer>ssl-1</sub-layer>
            <sub-layer>ssl-2</sub-layer>
          </service-sub-layer>
        </incoming>
        <outgoing>
          <interface>
            <outgoing-interface>eth2</outgoing-interface>
            <mpls-label-stack>
              <entry>
                <id>0</id>
                <label>10000</label>
              </entry>
            </mpls-label-stack>
           </interface>
        </outgoing>
      </sub-layer>
      </forwarding>
  </detnet>
```

Figure 6: Example B-1 DetNet configuration Forwarding Layer Aggregation

B.3. Example B-2 JSON Service Aggregation Configuration

Figure 7 illustrates DetNet service sub-layer flows 1 and 2 are aggregated into a service sub-layer of an aggregated flow. Multiple DetNet flows with the same requirements for the same destination are aggregated into a single aggregated DetNet flow, and service protection and resource allocation are performed by an aggregated DetNet flow service sub-layer and forwarding sub-layer. A diagram illustrating this case is shown and then the corresponding JSON operational data for node Ingress 1 follows.

Please consult the PDF or HTML versions for the Case B-2 Diagram.

Figure 7: Case B-2 Example Service Aggregation

Figure 8 contains the operational JSON configuration for the ingress aggregation node illustrated in Figure 7. In this example, service sub-layer ssl-1 for DetNet flow DN1 and ssl-2 for flow DN2 aggregate at service sub-layer Detet flow asl-1. In this example an aggregation service sub-layer asl-1 is created to aggregated ssl-1 and ssl2 and that label is encapsulated in a separate forwarding sub-layer afl-1 with MPLS labels.

```
{
  "ietf-detnet:detnet": {
    "traffic-profile": [
      {
        "name": "1",
        "traffic-requirements": {
          "min-bandwidth": "100000000",
          "max-latency": 100000000,
          "max-latency-variation": 20000000,
          "max-loss": "0.0000001",
          "max-consecutive-loss-tolerance": 5,
          "max-misordering": 0
        },
        "traffic-spec": {
          "interval": 125,
          "max-pkts-per-interval": 10,
          "max-payload-size": 1500
        },
        "member-app-flow": [
          "app-1",
          "app-2"
        ]
      },
      {
        "name": "2",
        "traffic-requirements": {
          "min-bandwidth": "100000000",
          "max-latency": 100000000,
          "max-latency-variation": 20000000,
          "max-loss": "0.0000001",
          "max-consecutive-loss-tolerance": 5,
          "max-misordering": 0
        },
        "member-svc-sublayer": [
          "ssl-1",
          "ssl-2"
        ]
      },
      {
        "name": "3",
        "traffic-requirements": {
          "min-bandwidth": "200000000",
          "max-latency": 100000000,
          "max-latency-variation": 20000000,
          "max-loss": "0.0000001",
          "max-consecutive-loss-tolerance": 5,
          "max-misordering": 0
        },
```

```
          "member-svc-sublayer": [
            "asl-1"
          ]
        },
        {
          "name": "4",
          "traffic-spec": {
            "interval": 125,
            "max-pkts-per-interval": 20,
            "max-payload-size": 1500
          },
          "member-fwd-sublayer": [
            "afl-1"
          ]
        }
      ],
      "app-flows": {
        "app-flow": [
          {
            "name": "app-1",
            "bidir-congruent": false,
            "outgoing-service": "ssl-1",
            "traffic-profile": "1",
            "ingress": {
              "app-flow-status": "ietf-detnet:ready",
              "interface": [
                "eth0"
              ],
              "ip-app-flow": {
                "src-ip-prefix": "192.0.2.1/32",
                "dest-ip-prefix": "192.0.2.8/32",
                "dscp": 6
              }
            }
          },
          {
            "name": "app-2",
            "bidir-congruent": false,
            "outgoing-service": "ssl-2",
            "traffic-profile": "1",
            "ingress": {
              "app-flow-status": "ietf-detnet:ready",
              "interface": [
                "eth0"
              ],
              "ip-app-flow": {
                "src-ip-prefix": "192.0.2.2/32",
                "dest-ip-prefix": "192.0.2.9/32",
```

```
                "dscp": 7
              }
            }
          }
        ]
      },
      "service": {
        "sub-layer": [
          {
            "name": "ssl-1",
            "service-rank": 10,
            "traffic-profile": "2",
            "service-protection": {
              "protection": "none",
              "sequence-number-length": "long-sn"
            },
            "operation": "initiation",
            "incoming": {
              "app-flow": {
                "flow": [
                  "app-1"
                ]
              }
            },
            "outgoing": {
              "service-sub-layer": {
                "aggregation-sub-layer": "asl-1",
                "service-label": {
                  "mpls-label-stack": {
                    "entry": [
                      {
                        "id": 0,
                        "label": 102
                      }
                    ]
                  }
                }
              }
            }
          },
          {
            "name": "ssl-2",
            "service-rank": 10,
            "traffic-profile": "2",
            "service-protection": {
              "protection": "none",
              "sequence-number-length": "long-sn"
            },
```

```
            "operation": "initiation",
            "incoming": {
              "app-flow": {
                "flow": [
                  "app-2"
                ]
              }
            },
            "outgoing": {
              "service-sub-layer": {
                "aggregation-sub-layer": "asl-1",
                "service-label": {
                  "mpls-label-stack": {
                    "entry": [
                      {
                        "id": 0,
                        "label": 105
                      }
                    ]
                  }
                }
              }
            }
          },
          {
            "name": "asl-1",
            "service-rank": 10,
            "traffic-profile": "3",
            "service-protection": {
              "protection": "none",
              "sequence-number-length": "long-sn"
            },
            "operation": "initiation",
            "incoming": {
              "service-aggregation": {
                "sub-layer": [
                  "ssl-1",
                  "ssl-2"
                ]
              }
            },
            "outgoing": {
              "forwarding-sub-layer": {
                "service-outgoing": [
                  {
                    "index": 0,
                    "mpls-label-stack": {
                      "entry": [
```

```
                        {
                          "id": 0,
                          "label": 1000
                        }
                      ]
                    },
                    "sub-layer": [
                      "afl-1"
                    ]
                  }
                ]
              }
            }
          }
        ]
      },
      "forwarding": {
        "sub-layer": [
          {
            "name": "afl-1",
            "traffic-profile": "4",
            "operation": "impose-and-forward",
            "incoming": {
              "service-sub-layer": {
                "sub-layer": [
                  "asl-1"
                ]
              }
            },
            "outgoing": {
              "interface": {
                "outgoing-interface": "eth2",
                "mpls-label-stack": {
                  "entry": [
                    {
                      "id": 0,
                      "label": 20000
                    }
                  ]
                }
              }
            }
          }
        ]
      }
    },
    "ietf-interfaces:interfaces": {
      "interface": [
```

```
        {
          "name": "eth0",
          "type": "iana-if-type:ethernetCsmacd",
          "oper-status": "up",
          "statistics": {
            "discontinuity-time": "2024-02-21T18:59:00-05:00"
          }
        },
        {
          "name": "eth2",
          "type": "iana-if-type:ethernetCsmacd",
          "oper-status": "up",
          "statistics": {
            "discontinuity-time": "2024-02-21T18:59:00-05:00"
          }
        }
      ]
    }
  }
```

Figure 8: Example B-2 DetNet Service Aggregation

B.4. Example C-1 JSON Relay Aggregation/Disaggregation Configuration

Figure 9 illustrates the relay node's forwarding sub-layer flows 1 and 2 aggregated into a single forwarding sub-layer. Service protection and resource allocation are performed by the corresponding service sub-layer and forwarding sub-layer of each flow. A diagram illustrating both aggregation and disaggregation is shown and then the corresponding JSON operational data follows.

Please consult the PDF or HTML versions for the Case C-1 Diagram.

Figure 9: Case C-1 Example Service Aggregation/Disaggregation

Figure 10 contains the operational JSON configuration for the ingress aggregation node illustrated in Figure 9. In this example, a relay performing aggregation at the forwarding sub-layer is illustrated. Two DetNet flows DN1 and DN2 are replicated at each service sub-layer. The two forwarding sub-layers for the upside path are aggregated at the forwarding sub-layer with label 20000 and the two forwarding sub-layers for the downside path are aggregated at the forwarding sub-layer with label 20001. Figure 11 contains the operational JSON configuration for the egress disaggregation node illustrated in Figure 9.

```
{
  "ietf-detnet:detnet": {
    "traffic-profile": [
      {
        "name": "pf-1",
        "traffic-requirements": {
          "min-bandwidth": "100000000",
          "max-latency": 100000000,
          "max-latency-variation": 10000000,
          "max-loss": "0.0000001",
          "max-consecutive-loss-tolerance": 5,
          "max-misordering": 0
        },
        "member-svc-sublayer": [
          "ssl-1",
          "ssl-2"
        ]
      },
      {
        "name": "pf-2",
        "traffic-spec": {
          "interval": 125,
          "max-pkts-per-interval": 2,
          "max-payload-size": 1518
        },
        "member-fwd-sublayer": [
          "afl-1",
          "afl-2"
        ]
      },
      {
        "name": "pf-3",
        "traffic-spec": {
          "interval": 125,
          "max-pkts-per-interval": 1,
          "max-payload-size": 1518
        },
        "member-fwd-sublayer": [
          "fsl-1",
          "fsl-2",
          "fsl-3",
          "fsl-4",
          "fsl-5",
          "fsl-6"
        ]
      }
    ],
    "service": {
```

```
        "sub-layer": [
          {
            "name": "ssl-1",
            "service-rank": 10,
            "traffic-profile": "pf-1",
            "service-protection": {
              "protection": "replication",
              "sequence-number-length": "long-sn"
            },
            "operation": "relay",
            "incoming": {
              "service-id": {
                "mpls-label-stack": {
                  "entry": [
                    {
                      "id": 0,
                      "label": 100
                    }
                  ]
                }
              }
            },
            "outgoing": {
              "forwarding-sub-layer": {
                "service-outgoing": [
                  {
                    "index": 0,
                    "mpls-label-stack": {
                      "entry": [
                        {
                          "id": 0,
                          "label": 101
                        }
                      ]
                    },
                    "sub-layer": [
                      "fsl-2",
                      "fsl-3"
                    ]
                  }
                ]
              }
            }
          },
          {
            "name": "ssl-2",
            "service-rank": 10,
            "traffic-profile": "pf-1",
```

```
            "service-protection": {
              "protection": "replication",
              "sequence-number-length": "long-sn"
            },
            "operation": "relay",
            "incoming": {
              "service-id": {
                "mpls-label-stack": {
                  "entry": [
                    {
                      "id": 0,
                      "label": 103
                    }
                  ]
                }
              }
            },
            "outgoing": {
              "forwarding-sub-layer": {
                "service-outgoing": [
                  {
                    "index": 0,
                    "mpls-label-stack": {
                      "entry": [
                        {
                          "id": 0,
                          "label": 104
                        }
                      ]
                    },
                    "sub-layer": [
                      "fsl-5",
                      "fsl-6"
                    ]
                  }
                ]
              }
            }
          }
        ]
      },
      "forwarding": {
        "sub-layer": [
          {
            "name": "fsl-1",
            "traffic-profile": "pf-3",
            "operation": "pop-and-lookup",
            "incoming": {
```

```
            "forwarding-id": {
              "interface": "eth0",
              "mpls-label-stack": {
                "entry": [
                  {
                    "id": 0,
                    "label": 10000
                  }
                ]
              }
            }
          },
          "outgoing": {
            "service-sub-layer": {
              "sub-layer": [
                "ssl-1"
              ]
            }
          }
        },
        {
          "name": "fsl-2",
          "traffic-profile": "pf-3",
          "operation": "impose-and-forward",
          "incoming": {
            "service-sub-layer": {
              "sub-layer": [
                "ssl-1"
              ]
            }
          },
          "outgoing": {
            "forwarding-sub-layer": {
              "aggregation-sub-layer": "afl-1",
              "forwarding-label": {
                "mpls-label-stack": {
                  "entry": [
                    {
                      "id": 0,
                      "label": 10003
                    }
                  ]
                }
              }
            }
          }
        },
        {
```

```
            "name": "fsl-3",
            "traffic-profile": "pf-3",
            "operation": "impose-and-forward",
            "incoming": {
              "service-sub-layer": {
                "sub-layer": [
                  "ssl-1"
                ]
              }
            },
            "outgoing": {
              "forwarding-sub-layer": {
                "aggregation-sub-layer": "afl-2",
                "forwarding-label": {
                  "mpls-label-stack": {
                    "entry": [
                      {
                        "id": 0,
                        "label": 10004
                      }
                    ]
                  }
                }
              }
            }
          },
          {
            "name": "fsl-4",
            "traffic-profile": "pf-3",
            "operation": "pop-and-lookup",
            "incoming": {
              "forwarding-id": {
                "interface": "eth1",
                "mpls-label-stack": {
                  "entry": [
                    {
                      "id": 0,
                      "label": 10006
                    }
                  ]
                }
              }
            },
            "outgoing": {
              "service-sub-layer": {
                "sub-layer": [
                  "ssl-2"
                ]
```

```
            }
          }
        },
        {
          "name": "fsl-5",
          "traffic-profile": "pf-3",
          "operation": "impose-and-forward",
          "incoming": {
            "service-sub-layer": {
              "sub-layer": [
                "ssl-2"
              ]
            }
          },
          "outgoing": {
            "forwarding-sub-layer": {
              "aggregation-sub-layer": "afl-1",
              "forwarding-label": {
                "mpls-label-stack": {
                  "entry": [
                    {
                      "id": 0,
                      "label": 10009
                    }
                  ]
                }
              }
            }
          }
        },
        {
          "name": "fsl-6",
          "traffic-profile": "pf-3",
          "operation": "impose-and-forward",
          "incoming": {
            "service-sub-layer": {
              "sub-layer": [
                "ssl-2"
              ]
            }
          },
          "outgoing": {
            "forwarding-sub-layer": {
              "aggregation-sub-layer": "afl-2",
              "forwarding-label": {
                "mpls-label-stack": {
                  "entry": [
                    {
```

```
                        "id": 0,
                        "label": 10010
                      }
                    ]
                  }
                }
              }
            }
          },
          {
            "name": "afl-1",
            "traffic-profile": "pf-2",
            "operation": "impose-and-forward",
            "incoming": {
              "forwarding-aggregation": {
                "sub-layer": [
                  "fsl-2",
                  "fsl-5"
                ]
              }
            },
            "outgoing": {
              "interface": {
                "outgoing-interface": "eth2",
                "mpls-label-stack": {
                  "entry": [
                    {
                      "id": 0,
                      "label": 20000
                    }
                  ]
                }
              }
            }
          },
          {
            "name": "afl-2",
            "traffic-profile": "pf-2",
            "operation": "impose-and-forward",
            "incoming": {
              "forwarding-aggregation": {
                "sub-layer": [
                  "fsl-3",
                  "fsl-6"
                ]
              }
            },
            "outgoing": {
```

```
              "interface": {
                "outgoing-interface": "eth3",
                "mpls-label-stack": {
                  "entry": [
                    {
                      "id": 0,
                      "label": 20001
                    }
                  ]
                }
              }
            }
          }
        ]
      }
    },
    "ietf-interfaces:interfaces": {
      "interface": [
        {
          "name": "eth0",
          "type": "iana-if-type:ethernetCsmacd",
          "oper-status": "up",
          "statistics": {
            "discontinuity-time": "2024-02-21T18:59:00-05:00"
          }
        },
        {
          "name": "eth1",
          "type": "iana-if-type:ethernetCsmacd",
          "oper-status": "up",
          "statistics": {
            "discontinuity-time": "2024-02-21T18:59:00-05:00"
          }
        },
        {
          "name": "eth2",
          "type": "iana-if-type:ethernetCsmacd",
          "oper-status": "up",
          "statistics": {
            "discontinuity-time": "2024-02-21T18:59:00-05:00"
          }
        },
        {
          "name": "eth3",
          "type": "iana-if-type:ethernetCsmacd",
          "oper-status": "up",
          "statistics": {
            "discontinuity-time": "2024-02-21T18:59:00-05:00"
```

```
          }
        }
      ]
    }
  }
```

Figure 10: Example C-1 DetNet Relay Service Aggregation

```
{
  "ietf-detnet:detnet": {
    "traffic-profile": [
      {
        "name": "pf-1",
        "traffic-requirements": {
          "min-bandwidth": "100000000",
          "max-latency": 100000000,
          "max-latency-variation": 10000000,
          "max-loss": "0.0000001",
          "max-consecutive-loss-tolerance": 5,
          "max-misordering": 0
        },
        "member-svc-sublayer": [
          "ssl-1",
          "ssl-2"
        ]
      },
      {
        "name": "pf-2",
        "traffic-spec": {
          "interval": 125,
          "max-pkts-per-interval": 2,
          "max-payload-size": 1518
        },
        "member-fwd-sublayer": [
          "afl-1",
          "afl-2"
        ]
      },
      {
        "name": "pf-3",
        "traffic-spec": {
          "interval": 125,
          "max-pkts-per-interval": 1,
          "max-payload-size": 1518
        },
        "member-fwd-sublayer": [
          "fsl-1",
          "fsl-2",
```

```
            "fsl-3",
            "fsl-4",
            "fsl-5",
            "fsl-6"
          ]
        }
      ],
      "service": {
        "sub-layer": [
          {
            "name": "ssl-1",
            "service-rank": 10,
            "traffic-profile": "pf-1",
            "service-protection": {
              "protection": "elimination",
              "sequence-number-length": "long-sn"
            },
            "operation": "relay",
            "incoming": {
              "service-id": {
                "mpls-label-stack": {
                  "entry": [
                    {
                      "id": 0,
                      "label": 101
                    }
                  ]
                }
              }
            },
            "outgoing": {
              "forwarding-sub-layer": {
                "service-outgoing": [
                  {
                    "index": 0,
                    "mpls-label-stack": {
                      "entry": [
                        {
                          "id": 0,
                          "label": 102
                        }
                      ]
                    },
                    "sub-layer": [
                      "fsl-3"
                    ]
                  }
                ]
```

```
              }
            }
          },
          {
            "name": "ssl-2",
            "service-rank": 10,
            "traffic-profile": "pf-1",
            "service-protection": {
              "protection": "elimination",
              "sequence-number-length": "long-sn"
            },
            "operation": "relay",
            "incoming": {
              "service-id": {
                "mpls-label-stack": {
                  "entry": [
                    {
                      "id": 0,
                      "label": 104
                    }
                  ]
                }
              }
            },
            "outgoing": {
              "forwarding-sub-layer": {
                "service-outgoing": [
                  {
                    "index": 0,
                    "mpls-label-stack": {
                      "entry": [
                        {
                          "id": 0,
                          "label": 105
                        }
                      ]
                    },
                    "sub-layer": [
                      "fsl-6"
                    ]
                  }
                ]
              }
            }
          }
        ]
      },
      "forwarding": {
```

```
        "sub-layer": [
          {
            "name": "afl-1",
            "traffic-profile": "pf-2",
            "operation": "pop-and-lookup",
            "incoming": {
              "forwarding-id": {
                "interface": "eth0",
                "mpls-label-stack": {
                  "entry": [
                    {
                      "id": 0,
                      "label": 20002
                    }
                  ]
                }
              }
            },
            "outgoing": {
              "forwarding-disaggregation": {
                "sub-layer": [
                  "fsl-1",
                  "fsl-4"
                ]
              }
            }
          },
          {
            "name": "afl-2",
            "traffic-profile": "pf-2",
            "operation": "pop-and-lookup",
            "incoming": {
              "forwarding-id": {
                "interface": "eth1",
                "mpls-label-stack": {
                  "entry": [
                    {
                      "id": 0,
                      "label": 20003
                    }
                  ]
                }
              }
            },
            "outgoing": {
              "forwarding-disaggregation": {
                "sub-layer": [
                  "fsl-2",
```

```
                "fsl-5"
              ]
            }
          }
        },
        {
          "name": "fsl-1",
          "traffic-profile": "pf-3",
          "operation": "pop-and-lookup",
          "incoming": {
            "forwarding-id": {
              "interface": "eth0",
              "mpls-label-stack": {
                "entry": [
                  {
                    "id": 0,
                    "label": 10003
                  }
                ]
              }
            }
          },
          "outgoing": {
            "service-sub-layer": {
              "sub-layer": [
                "ssl-1"
              ]
            }
          }
        },
        {
          "name": "fsl-2",
          "traffic-profile": "pf-3",
          "operation": "pop-and-lookup",
          "incoming": {
            "forwarding-id": {
              "interface": "eth1",
              "mpls-label-stack": {
                "entry": [
                  {
                    "id": 0,
                    "label": 10004
                  }
                ]
              }
            }
          },
          "outgoing": {
```

```
            "service-sub-layer": {
              "sub-layer": [
                "ssl-1"
              ]
            }
          }
        },
        {
          "name": "fsl-3",
          "traffic-profile": "pf-3",
          "operation": "impose-and-forward",
          "incoming": {
            "service-sub-layer": {
              "sub-layer": [
                "ssl-1"
              ]
            }
          },
          "outgoing": {
            "interface": {
              "outgoing-interface": "eth2",
              "mpls-label-stack": {
                "entry": [
                  {
                    "id": 0,
                    "label": 10005
                  }
                ]
              }
            }
          }
        },
        {
          "name": "fsl-4",
          "traffic-profile": "pf-3",
          "operation": "pop-and-lookup",
          "incoming": {
            "forwarding-id": {
              "interface": "eth0",
              "mpls-label-stack": {
                "entry": [
                  {
                    "id": 0,
                    "label": 10009
                  }
                ]
              }
            }
```

```
        },
        "outgoing": {
          "service-sub-layer": {
            "sub-layer": [
              "ssl-2"
            ]
          }
        }
      },
      {
        "name": "fsl-5",
        "traffic-profile": "pf-3",
        "operation": "pop-and-lookup",
        "incoming": {
          "forwarding-id": {
            "interface": "eth1",
            "mpls-label-stack": {
              "entry": [
                {
                  "id": 0,
                  "label": 10010
                }
              ]
            }
          }
        },
        "outgoing": {
          "service-sub-layer": {
            "sub-layer": [
              "ssl-2"
            ]
          }
        }
      },
      {
        "name": "fsl-6",
        "traffic-profile": "pf-3",
        "operation": "impose-and-forward",
        "incoming": {
          "service-sub-layer": {
            "sub-layer": [
              "ssl-2"
            ]
          }
        },
        "outgoing": {
          "interface": {
            "outgoing-interface": "eth3",
```

```
                "mpls-label-stack": {
                  "entry": [
                    {
                      "id": 0,
                      "label": 10011
                    }
                  ]
                }
              }
            }
          }
        ]
      }
    },
    "ietf-interfaces:interfaces": {
      "interface": [
        {
          "name": "eth0",
          "type": "iana-if-type:ethernetCsmacd",
          "oper-status": "up",
          "statistics": {
            "discontinuity-time": "2024-02-21T18:59:00-05:00"
          }
        },
        {
          "name": "eth1",
          "type": "iana-if-type:ethernetCsmacd",
          "oper-status": "up",
          "statistics": {
            "discontinuity-time": "2024-02-21T18:59:00-05:00"
          }
        },
        {
          "name": "eth2",
          "type": "iana-if-type:ethernetCsmacd",
          "oper-status": "up",
          "statistics": {
            "discontinuity-time": "2024-02-21T18:59:00-05:00"
          }
        },
        {
          "name": "eth3",
          "type": "iana-if-type:ethernetCsmacd",
          "oper-status": "up",
          "statistics": {
            "discontinuity-time": "2024-02-21T18:59:00-05:00"
          }
        }
```

```
      ]
    }
  }
```

Figure 11: Example C-1 DetNet Relay Service Disaggregation

B.5. Example C-2 JSON Relay Aggregation/Disaggregation Service Sub-Layer

Figure 12 illustrates the DetNet relay node service sub-layer flows 1 and 2 aggregated into a single forwarding sub-layer. Service protection is performed by the corresponding service sub-layer of each flow and resource allocation is performed by an aggregated forwarding sub-layer for all aggregated flows. A diagram illustrating both aggregation and disaggregation is shown and then the corresponding JSON operational data follows.

Please consult the PDF or HTML versions for the Case C-2 Diagram.

Figure 12: Case C-2 Example Service Aggregation/Disaggregation

Figure 13 contains the operational JSON configuration for the ingress aggregation node illustrated in Figure 12. In this example, a relay performing aggregation at the forwarding sub-layer is illustrated. Two DetNet flows DN1 and DN2 are replicated at each service sub-layer, and the replicated each service sub-layer for the upside path is aggregated at the single forwarding sub-layer with MPLS label 20000, and the replicated each service sub-layer for the downside path is aggregated at the forwarding sub-layer with MPLS label 20001. Figure 14 contains the operational JSON configuration for the egress disaggregation node illustrated in Figure 12.

```
{
  "ietf-detnet:detnet": {
    "traffic-profile": [
      {
        "name": "pf-1",
        "traffic-requirements": {
          "min-bandwidth": "100000000",
          "max-latency": 100000000,
          "max-latency-variation": 10000000,
          "max-loss": "0.0000001",
          "max-consecutive-loss-tolerance": 5,
          "max-misordering": 0
        },
        "member-svc-sublayer": [
          "ssl-1",
          "ssl-2"
```

```
          ]
        },
        {
          "name": "pf-2",
          "traffic-spec": {
            "interval": 125,
            "max-pkts-per-interval": 1,
            "max-payload-size": 1518
          },
          "member-fwd-sublayer": [
            "fsl-1",
            "fsl-2"
          ]
        },
        {
          "name": "pf-3",
          "traffic-spec": {
            "interval": 125,
            "max-pkts-per-interval": 2,
            "max-payload-size": 1518
          },
          "member-fwd-sublayer": [
            "afl-1",
            "afl-2"
          ]
        }
      ],
      "service": {
        "sub-layer": [
          {
            "name": "ssl-1",
            "service-rank": 10,
            "traffic-profile": "pf-1",
            "service-protection": {
              "protection": "replication",
              "sequence-number-length": "long-sn"
            },
            "operation": "relay",
            "incoming": {
              "service-id": {
                "mpls-label-stack": {
                  "entry": [
                    {
                      "id": 0,
                      "label": 100
                    }
                  ]
                }
```

```
            }
          },
          "outgoing": {
            "forwarding-sub-layer": {
              "service-outgoing": [
                {
                  "index": 0,
                  "mpls-label-stack": {
                    "entry": [
                      {
                        "id": 0,
                        "label": 101
                      }
                    ]
                  },
                  "sub-layer": [
                    "afl-1",
                    "afl-2"
                  ]
                }
              ]
            }
          }
        },
        {
          "name": "ssl-2",
          "service-rank": 10,
          "traffic-profile": "pf-1",
          "service-protection": {
            "protection": "replication",
            "sequence-number-length": "long-sn"
          },
          "operation": "relay",
          "incoming": {
            "service-id": {
              "mpls-label-stack": {
                "entry": [
                  {
                    "id": 0,
                    "label": 103
                  }
                ]
              }
            }
          },
          "outgoing": {
            "forwarding-sub-layer": {
              "service-outgoing": [
```

```
                {
                  "index": 0,
                  "mpls-label-stack": {
                    "entry": [
                      {
                        "id": 0,
                        "label": 104
                      }
                    ]
                  },
                  "sub-layer": [
                    "afl-1",
                    "afl-2"
                  ]
                }
              ]
            }
          }
        }
      ]
    },
    "forwarding": {
      "sub-layer": [
        {
          "name": "fsl-1",
          "traffic-profile": "pf-2",
          "operation": "pop-and-lookup",
          "incoming": {
            "forwarding-id": {
              "interface": "eth0",
              "mpls-label-stack": {
                "entry": [
                  {
                    "id": 0,
                    "label": 10000
                  }
                ]
              }
            }
          },
          "outgoing": {
            "service-sub-layer": {
              "sub-layer": [
                "ssl-1"
              ]
            }
          }
        },
```

```
          {
            "name": "fsl-2",
            "traffic-profile": "pf-2",
            "operation": "pop-and-lookup",
            "incoming": {
              "forwarding-id": {
                "interface": "eth1",
                "mpls-label-stack": {
                  "entry": [
                    {
                      "id": 0,
                      "label": 10006
                    }
                  ]
                }
              }
            },
            "outgoing": {
              "service-sub-layer": {
                "sub-layer": [
                  "ssl-2"
                ]
              }
            }
          },
          {
            "name": "afl-1",
            "traffic-profile": "pf-3",
            "operation": "impose-and-forward",
            "incoming": {
              "service-sub-layer": {
                "sub-layer": [
                  "ssl-1",
                  "ssl-2"
                ]
              }
            },
            "outgoing": {
              "interface": {
                "outgoing-interface": "eth2",
                "mpls-label-stack": {
                  "entry": [
                    {
                      "id": 0,
                      "label": 20000
                    }
                  ]
                }
```

```
              }
            }
          },
          {
            "name": "afl-2",
            "traffic-profile": "pf-3",
            "operation": "impose-and-forward",
            "incoming": {
              "service-sub-layer": {
                "sub-layer": [
                  "ssl-1",
                  "ssl-2"
                ]
              }
            },
            "outgoing": {
              "interface": {
                "outgoing-interface": "eth3",
                "mpls-label-stack": {
                  "entry": [
                    {
                      "id": 0,
                      "label": 20001
                    }
                  ]
                }
              }
            }
          }
        ]
      }
    },
    "ietf-interfaces:interfaces": {
      "interface": [
        {
          "name": "eth0",
          "type": "iana-if-type:ethernetCsmacd",
          "oper-status": "up",
          "statistics": {
            "discontinuity-time": "2024-02-21T18:59:00-05:00"
          }
        },
        {
          "name": "eth1",
          "type": "iana-if-type:ethernetCsmacd",
          "oper-status": "up",
          "statistics": {
            "discontinuity-time": "2024-02-21T18:59:00-05:00"
```

```
          }
        },
        {
          "name": "eth2",
          "type": "iana-if-type:ethernetCsmacd",
          "oper-status": "up",
          "statistics": {
            "discontinuity-time": "2024-02-21T18:59:00-05:00"
          }
        },
        {
          "name": "eth3",
          "type": "iana-if-type:ethernetCsmacd",
          "oper-status": "up",
          "statistics": {
            "discontinuity-time": "2024-02-21T18:59:00-05:00"
          }
        }
      ]
    }
  }
```

Figure 13: Example C-2 DetNet Relay Aggregation Service Sub-Layer

```
{
  "ietf-detnet:detnet": {
    "traffic-profile": [
      {
        "name": "pf-1",
        "traffic-requirements": {
          "min-bandwidth": "100000000",
          "max-latency": 100000000,
          "max-latency-variation": 10000000,
          "max-loss": "0.0000001",
          "max-consecutive-loss-tolerance": 5,
          "max-misordering": 0
        },
        "member-svc-sublayer": [
          "ssl-1",
          "ssl-2"
        ]
      },
      {
        "name": "pf-2",
        "traffic-spec": {
          "interval": 125,
          "max-pkts-per-interval": 1,
          "max-payload-size": 1518
```

```
          },
          "member-fwd-sublayer": [
            "fsl-1",
            "fsl-2"
          ]
        },
        {
          "name": "pf-3",
          "traffic-spec": {
            "interval": 125,
            "max-pkts-per-interval": 2,
            "max-payload-size": 1518
          },
          "member-fwd-sublayer": [
            "afl-1",
            "afl-2"
          ]
        }
      ],
      "service": {
        "sub-layer": [
          {
            "name": "ssl-1",
            "service-rank": 10,
            "traffic-profile": "pf-1",
            "service-protection": {
              "protection": "elimination",
              "sequence-number-length": "long-sn"
            },
            "operation": "relay",
            "incoming": {
              "service-id": {
                "mpls-label-stack": {
                  "entry": [
                    {
                      "id": 0,
                      "label": 101
                    }
                  ]
                }
              }
            },
            "outgoing": {
              "forwarding-sub-layer": {
                "service-outgoing": [
                  {
                    "index": 0,
                    "mpls-label-stack": {
```

```
                    "entry": [
                      {
                        "id": 0,
                        "label": 102
                      }
                    ]
                  },
                  "sub-layer": [
                    "fsl-1"
                  ]
                }
              ]
            }
          }
        },
        {
          "name": "ssl-2",
          "service-rank": 10,
          "traffic-profile": "pf-1",
          "service-protection": {
            "protection": "elimination",
            "sequence-number-length": "long-sn"
          },
          "operation": "relay",
          "incoming": {
            "service-id": {
              "mpls-label-stack": {
                "entry": [
                  {
                    "id": 0,
                    "label": 104
                  }
                ]
              }
            }
          },
          "outgoing": {
            "forwarding-sub-layer": {
              "service-outgoing": [
                {
                  "index": 0,
                  "mpls-label-stack": {
                    "entry": [
                      {
                        "id": 0,
                        "label": 105
                      }
                    ]
```

```
                    },
                    "sub-layer": [
                      "fsl-2"
                    ]
                  }
                ]
              }
            }
          }
        ]
      },
      "forwarding": {
        "sub-layer": [
          {
            "name": "afl-1",
            "traffic-profile": "pf-3",
            "operation": "pop-and-lookup",
            "incoming": {
              "forwarding-id": {
                "interface": "eth0",
                "mpls-label-stack": {
                  "entry": [
                    {
                      "id": 0,
                      "label": 20002
                    }
                  ]
                }
              }
            },
            "outgoing": {
              "service-sub-layer": {
                "sub-layer": [
                  "ssl-1",
                  "ssl-2"
                ]
              }
            }
          },
          {
            "name": "afl-2",
            "traffic-profile": "pf-3",
            "operation": "pop-and-lookup",
            "incoming": {
              "forwarding-id": {
                "interface": "eth1",
                "mpls-label-stack": {
                  "entry": [
```

```
                    {
                      "id": 0,
                      "label": 20003
                    }
                  ]
                }
              }
            },
            "outgoing": {
              "service-sub-layer": {
                "sub-layer": [
                  "ssl-1",
                  "ssl-2"
                ]
              }
            }
          },
          {
            "name": "fsl-1",
            "traffic-profile": "pf-2",
            "operation": "impose-and-forward",
            "incoming": {
              "service-sub-layer": {
                "sub-layer": [
                  "ssl-1"
                ]
              }
            },
            "outgoing": {
              "interface": {
                "outgoing-interface": "eth2",
                "mpls-label-stack": {
                  "entry": [
                    {
                      "id": 0,
                      "label": 10005
                    }
                  ]
                }
              }
            }
          },
          {
            "name": "fsl-2",
            "traffic-profile": "pf-2",
            "operation": "impose-and-forward",
            "incoming": {
              "service-sub-layer": {
```

```
                "sub-layer": [
                  "ssl-2"
                ]
              }
            },
            "outgoing": {
              "interface": {
                "outgoing-interface": "eth3",
                "mpls-label-stack": {
                  "entry": [
                    {
                      "id": 0,
                      "label": 10011
                    }
                  ]
                }
              }
            }
          }
        ]
      }
    },
    "ietf-interfaces:interfaces": {
      "interface": [
        {
          "name": "eth0",
          "type": "iana-if-type:ethernetCsmacd",
          "oper-status": "up",
          "statistics": {
            "discontinuity-time": "2024-02-21T18:59:00-05:00"
          }
        },
        {
          "name": "eth1",
          "type": "iana-if-type:ethernetCsmacd",
          "oper-status": "up",
          "statistics": {
            "discontinuity-time": "2024-02-21T18:59:00-05:00"
          }
        },
        {
          "name": "eth2",
          "type": "iana-if-type:ethernetCsmacd",
          "oper-status": "up",
          "statistics": {
            "discontinuity-time": "2024-02-21T18:59:00-05:00"
          }
        },
```

```
        {
          "name": "eth3",
          "type": "iana-if-type:ethernetCsmacd",
          "oper-status": "up",
          "statistics": {
            "discontinuity-time": "2024-02-21T18:59:00-05:00"
          }
        }
      ]
    }
  }
```

Figure 14: Example C-2 DetNet Relay Disaggregation Service Sub-Layer

B.6. Example C-3 JSON Relay Service Sub-Layer Aggregation/ Disaggregation

Figure 15 illustrates the DetNet relay node service sub-layer flows 1 and 2 aggregated into a service sub-layer flow. Multiple DetNet flows with the same requirements which can use the same path are aggregated into a single aggregated DetNet flow, and service protection and resource allocation are performed by the service sub-layer and forwarding sub-layer of aggregated DetNet flow. A diagram illustrating both aggregation and disaggregation is shown and then the corresponding JSON operational data follows.

Please consult the PDF or HTML versions for the Case C-3 Diagram.

Figure 15: Case C-3 Example Service Aggregation/Disaggregation

Figure 16 contains the operational JSON configuration for the ingress aggregation node illustrated in Figure 15. In this example a relay performing aggregation at the service sub-layer is illustrated. Two DetNet flows DN1 and DN2 are relayed at each service sub-layer with MPLS labels 101 and 104 respectively, and each service sub-layer is aggregated at a single service sub-layer flow and replicated. Figure 17 contains the operational JSON configuration for the egress disaggregation node illustrated in Figure 15.

```
{
  "ietf-detnet:detnet": {
    "traffic-profile": [
      {
        "name": "pf-1",
        "traffic-requirements": {
          "min-bandwidth": "100000000",
          "max-latency": 100000000,
          "max-latency-variation": 10000000,
```

```
            "max-loss": "0.0000001",
            "max-consecutive-loss-tolerance": 5,
            "max-misordering": 0
          },
          "member-svc-sublayer": [
            "ssl-1",
            "ssl-2"
          ]
        },
        {
          "name": "pf-2",
          "traffic-requirements": {
            "min-bandwidth": "200000000",
            "max-latency": 100000000,
            "max-latency-variation": 10000000,
            "max-loss": "0.0000001",
            "max-consecutive-loss-tolerance": 5,
            "max-misordering": 0
          },
          "member-svc-sublayer": [
            "asl-1"
          ]
        },
        {
          "name": "pf-3",
          "traffic-spec": {
            "interval": 125,
            "max-pkts-per-interval": 1,
            "max-payload-size": 1518
          },
          "member-fwd-sublayer": [
            "fsl-1",
            "fsl-2"
          ]
        },
        {
          "name": "pf-4",
          "traffic-spec": {
            "interval": 125,
            "max-pkts-per-interval": 2,
            "max-payload-size": 1518
          },
          "member-fwd-sublayer": [
            "fsl-3",
            "fsl-4"
          ]
        }
      ],
```

```
      "service": {
        "sub-layer": [
          {
            "name": "ssl-1",
            "service-rank": 10,
            "traffic-profile": "pf-1",
            "service-protection": {
              "protection": "none",
              "sequence-number-length": "long-sn"
            },
            "operation": "relay",
            "incoming": {
              "service-id": {
                "mpls-label-stack": {
                  "entry": [
                    {
                      "id": 0,
                      "label": 100
                    }
                  ]
                }
              }
            },
            "outgoing": {
              "service-sub-layer": {
                "aggregation-sub-layer": "asl-1",
                "service-label": {
                  "mpls-label-stack": {
                    "entry": [
                      {
                        "id": 0,
                        "label": 101
                      }
                    ]
                  }
                }
              }
            }
          },
          {
            "name": "ssl-2",
            "service-rank": 10,
            "traffic-profile": "pf-1",
            "service-protection": {
              "protection": "none",
              "sequence-number-length": "long-sn"
            },
            "operation": "relay",
```

```
            "incoming": {
              "service-id": {
                "mpls-label-stack": {
                  "entry": [
                    {
                      "id": 0,
                      "label": 103
                    }
                  ]
                }
              }
            },
            "outgoing": {
              "service-sub-layer": {
                "aggregation-sub-layer": "asl-1",
                "service-label": {
                  "mpls-label-stack": {
                    "entry": [
                      {
                        "id": 0,
                        "label": 104
                      }
                    ]
                  }
                }
              }
            }
          },
          {
            "name": "asl-1",
            "service-rank": 10,
            "traffic-profile": "pf-2",
            "service-protection": {
              "protection": "replication",
              "sequence-number-length": "long-sn"
            },
            "operation": "initiation",
            "incoming": {
              "service-aggregation": {
                "sub-layer": [
                  "ssl-1",
                  "ssl-2"
                ]
              }
            },
            "outgoing": {
              "forwarding-sub-layer": {
                "service-outgoing": [
```

```
                {
                  "index": 0,
                  "mpls-label-stack": {
                    "entry": [
                      {
                        "id": 0,
                        "label": 1000
                      }
                    ]
                  },
                  "sub-layer": [
                    "fsl-3",
                    "fsl-4"
                  ]
                }
              ]
            }
          }
        }
      ]
    },
    "forwarding": {
      "sub-layer": [
        {
          "name": "fsl-1",
          "traffic-profile": "pf-3",
          "operation": "pop-and-lookup",
          "incoming": {
            "forwarding-id": {
              "interface": "eth0",
              "mpls-label-stack": {
                "entry": [
                  {
                    "id": 0,
                    "label": 10000
                  }
                ]
              }
            }
          },
          "outgoing": {
            "service-sub-layer": {
              "sub-layer": [
                "ssl-1"
              ]
            }
          }
        },
```

```
          {
            "name": "fsl-2",
            "traffic-profile": "pf-3",
            "operation": "pop-and-lookup",
            "incoming": {
              "forwarding-id": {
                "interface": "eth1",
                "mpls-label-stack": {
                  "entry": [
                    {
                      "id": 0,
                      "label": 10006
                    }
                  ]
                }
              }
            },
            "outgoing": {
              "service-sub-layer": {
                "sub-layer": [
                  "ssl-2"
                ]
              }
            }
          },
          {
            "name": "fsl-3",
            "traffic-profile": "pf-4",
            "operation": "impose-and-forward",
            "incoming": {
              "service-sub-layer": {
                "sub-layer": [
                  "asl-1"
                ]
              }
            },
            "outgoing": {
              "interface": {
                "outgoing-interface": "eth2",
                "mpls-label-stack": {
                  "entry": [
                    {
                      "id": 0,
                      "label": 20000
                    }
                  ]
                }
              }
```

```
            }
          },
          {
            "name": "fsl-4",
            "traffic-profile": "pf-4",
            "operation": "impose-and-forward",
            "incoming": {
              "service-sub-layer": {
                "sub-layer": [
                  "asl-1"
                ]
              }
            },
            "outgoing": {
              "interface": {
                "outgoing-interface": "eth3",
                "mpls-label-stack": {
                  "entry": [
                    {
                      "id": 0,
                      "label": 20001
                    }
                  ]
                }
              }
            }
          }
        ]
      }
    },
    "ietf-interfaces:interfaces": {
      "interface": [
        {
          "name": "eth0",
          "type": "iana-if-type:ethernetCsmacd",
          "oper-status": "up",
          "statistics": {
            "discontinuity-time": "2024-02-21T18:59:00-05:00"
          }
        },
        {
          "name": "eth1",
          "type": "iana-if-type:ethernetCsmacd",
          "oper-status": "up",
          "statistics": {
            "discontinuity-time": "2024-02-21T18:59:00-05:00"
          }
        },
```

```
        {
          "name": "eth2",
          "type": "iana-if-type:ethernetCsmacd",
          "oper-status": "up",
          "statistics": {
            "discontinuity-time": "2024-02-21T18:59:00-05:00"
          }
        },
        {
          "name": "eth3",
          "type": "iana-if-type:ethernetCsmacd",
          "oper-status": "up",
          "statistics": {
            "discontinuity-time": "2024-02-21T18:59:00-05:00"
          }
        }
      ]
    }
  }
```

Figure 16: Example C-3 DetNet Relay Service Sub-Layer Aggregation

```
{
  "ietf-detnet:detnet": {
    "traffic-profile": [
      {
        "name": "pf-1",
        "traffic-requirements": {
          "min-bandwidth": "100000000",
          "max-latency": 100000000,
          "max-latency-variation": 10000000,
          "max-loss": "0.0000001",
          "max-consecutive-loss-tolerance": 5,
          "max-misordering": 0
        },
        "member-svc-sublayer": [
          "ssl-1",
          "ssl-2"
        ]
      },
      {
        "name": "pf-2",
        "traffic-requirements": {
          "min-bandwidth": "200000000",
          "max-latency": 100000000,
          "max-latency-variation": 10000000,
          "max-loss": "0.0000001",
          "max-consecutive-loss-tolerance": 5,
```

```
            "max-misordering": 0
          },
          "member-svc-sublayer": [
            "asl-1"
          ]
        },
        {
          "name": "pf-3",
          "traffic-spec": {
            "interval": 125,
            "max-pkts-per-interval": 1,
            "max-payload-size": 1518
          },
          "member-fwd-sublayer": [
            "fsl-3",
            "fsl-4"
          ]
        },
        {
          "name": "pf-4",
          "traffic-spec": {
            "interval": 125,
            "max-pkts-per-interval": 2,
            "max-payload-size": 1518
          },
          "member-fwd-sublayer": [
            "fsl-1",
            "fsl-2"
          ]
        }
      ],
      "service": {
        "sub-layer": [
          {
            "name": "ssl-1",
            "service-rank": 10,
            "traffic-profile": "pf-1",
            "service-protection": {
              "protection": "none",
              "sequence-number-length": "long-sn"
            },
            "operation": "relay",
            "incoming": {
              "service-id": {
                "mpls-label-stack": {
                  "entry": [
                    {
                      "id": 0,
```

```
                    "label": 101
                  }
                ]
              }
            }
          },
          "outgoing": {
            "forwarding-sub-layer": {
              "service-outgoing": [
                {
                  "index": 0,
                  "mpls-label-stack": {
                    "entry": [
                      {
                        "id": 0,
                        "label": 102
                      }
                    ]
                  },
                  "sub-layer": [
                    "fsl-3"
                  ]
                }
              ]
            }
          }
        },
        {
          "name": "ssl-2",
          "service-rank": 10,
          "traffic-profile": "pf-1",
          "service-protection": {
            "protection": "none",
            "sequence-number-length": "long-sn"
          },
          "operation": "relay",
          "incoming": {
            "service-id": {
              "mpls-label-stack": {
                "entry": [
                  {
                    "id": 0,
                    "label": 104
                  }
                ]
              }
            }
          },
```

```
            "outgoing": {
              "forwarding-sub-layer": {
                "service-outgoing": [
                  {
                    "index": 0,
                    "mpls-label-stack": {
                      "entry": [
                        {
                          "id": 0,
                          "label": 105
                        }
                      ]
                    },
                    "sub-layer": [
                      "fsl-4"
                    ]
                  }
                ]
              }
            }
          },
          {
            "name": "asl-1",
            "service-rank": 10,
            "traffic-profile": "pf-2",
            "service-protection": {
              "protection": "elimination",
              "sequence-number-length": "long-sn"
            },
            "operation": "termination",
            "incoming": {
              "service-id": {
                "mpls-label-stack": {
                  "entry": [
                    {
                      "id": 0,
                      "label": 1000
                    }
                  ]
                }
              }
            },
            "outgoing": {
              "service-disaggregation": {
                "sub-layer": [
                  "ssl-1",
                  "ssl-2"
                ]
```

```
              }
            }
          }
        ]
      },
      "forwarding": {
        "sub-layer": [
          {
            "name": "fsl-1",
            "traffic-profile": "pf-4",
            "operation": "pop-and-lookup",
            "incoming": {
              "forwarding-id": {
                "interface": "eth0",
                "mpls-label-stack": {
                  "entry": [
                    {
                      "id": 0,
                      "label": 20002
                    }
                  ]
                }
              }
            },
            "outgoing": {
              "service-sub-layer": {
                "sub-layer": [
                  "asl-1"
                ]
              }
            }
          },
          {
            "name": "fsl-2",
            "traffic-profile": "pf-4",
            "operation": "pop-and-lookup",
            "incoming": {
              "forwarding-id": {
                "interface": "eth1",
                "mpls-label-stack": {
                  "entry": [
                    {
                      "id": 0,
                      "label": 20003
                    }
                  ]
                }
              }
            }
```

```
            },
            "outgoing": {
              "service-sub-layer": {
                "sub-layer": [
                  "asl-1"
                ]
              }
            }
          },
          {
            "name": "fsl-3",
            "traffic-profile": "pf-3",
            "operation": "impose-and-forward",
            "incoming": {
              "service-sub-layer": {
                "sub-layer": [
                  "ssl-1"
                ]
              }
            },
            "outgoing": {
              "interface": {
                "outgoing-interface": "eth2",
                "mpls-label-stack": {
                  "entry": [
                    {
                      "id": 0,
                      "label": 10005
                    }
                  ]
                }
              }
            }
          },
          {
            "name": "fsl-4",
            "traffic-profile": "pf-3",
            "operation": "impose-and-forward",
            "incoming": {
              "service-sub-layer": {
                "sub-layer": [
                  "ssl-2"
                ]
              }
            },
            "outgoing": {
              "interface": {
                "outgoing-interface": "eth3",
```

```
                "mpls-label-stack": {
                  "entry": [
                    {
                      "id": 0,
                      "label": 10011
                    }
                  ]
                }
              }
            }
          }
        ]
      }
    },
    "ietf-interfaces:interfaces": {
      "interface": [
        {
          "name": "eth0",
          "type": "iana-if-type:ethernetCsmacd",
          "oper-status": "up",
          "statistics": {
            "discontinuity-time": "2024-02-21T18:59:00-05:00"
          }
        },
        {
          "name": "eth1",
          "type": "iana-if-type:ethernetCsmacd",
          "oper-status": "up",
          "statistics": {
            "discontinuity-time": "2024-02-21T18:59:00-05:00"
          }
        },
        {
          "name": "eth2",
          "type": "iana-if-type:ethernetCsmacd",
          "oper-status": "up",
          "statistics": {
            "discontinuity-time": "2024-02-21T18:59:00-05:00"
          }
        },
        {
          "name": "eth3",
          "type": "iana-if-type:ethernetCsmacd",
          "oper-status": "up",
          "statistics": {
            "discontinuity-time": "2024-02-21T18:59:00-05:00"
          }
        }
```

```
      ]
    }
  }
```

Figure 17: Example C-3 DetNet Relay Service Sub-Layer Disaggregation

B.7. Example C-4 JSON Relay Service Sub-Layer Aggregation/ Disaggregation

Figure 18 illustrates the relay node DetNet forwarding sub-layer flows 1 and 2 aggregated into a service sub-layer DetNet flow. Multiple DetNet flows with the same requirements which can use the same path are aggregated into a single aggregated DetNet flow. Service protection is performed by the service sub-layer of the aggregated DetNet flow and resource allocation is performed by the forwarding sub-layer of each aggregated DetNet flow. A diagram illustrating both aggregation and disaggregation is shown and then the corresponding JSON operational data follows.

Please consult the PDF or HTML versions for the Case C-4 Diagram

Figure 18: Case C-4 Example Service Aggregation/Disaggregation

Figure 19 contains the operational JSON configuration for the ingress aggregation node illustrated in Figure 18. In this example, a relay performing aggregation at the service sub-layer is illustrated. Two DetNet flows DN1 and DN2 are relayed at each service sub-layer. The two DetNet forwarding sub-layer flows with MPLS labels 20004 and 20005 are aggregated at the single service sub-layer DetNet flow and then replicated. Figure 20 contains the operational JSON configuration for the egress disaggregation node illustrated in Figure 18.

```
{
  "ietf-detnet:detnet": {
    "traffic-profile": [
      {
        "name": "pf-1",
        "traffic-requirements": {
          "min-bandwidth": "100000000",
          "max-latency": 100000000,
          "max-latency-variation": 10000000,
          "max-loss": "0.0000001",
          "max-consecutive-loss-tolerance": 5,
          "max-misordering": 0
        },
        "member-svc-sublayer": [
          "ssl-1",
```

```
            "ssl-2"
          ]
        },
        {
          "name": "pf-2",
          "traffic-requirements": {
            "min-bandwidth": "200000000",
            "max-latency": 100000000,
            "max-latency-variation": 10000000,
            "max-loss": "0.0000001",
            "max-consecutive-loss-tolerance": 5,
            "max-misordering": 0
          },
          "member-svc-sublayer": [
            "asl-1"
          ]
        },
        {
          "name": "pf-3",
          "traffic-spec": {
            "interval": 125,
            "max-pkts-per-interval": 1,
            "max-payload-size": 1518
          },
          "member-fwd-sublayer": [
            "fsl-1",
            "fsl-2",
            "fsl-3",
            "fsl-4"
          ]
        },
        {
          "name": "pf-4",
          "traffic-spec": {
            "interval": 125,
            "max-pkts-per-interval": 2,
            "max-payload-size": 1518
          },
          "member-fwd-sublayer": [
            "fsl-5",
            "fsl-6"
          ]
        }
      ],
      "service": {
        "sub-layer": [
          {
            "name": "ssl-1",
```

```
            "service-rank": 10,
            "traffic-profile": "pf-1",
            "service-protection": {
              "protection": "none",
              "sequence-number-length": "long-sn"
            },
            "operation": "relay",
            "incoming": {
              "service-id": {
                "mpls-label-stack": {
                  "entry": [
                    {
                      "id": 0,
                      "label": 100
                    }
                  ]
                }
              }
            },
            "outgoing": {
              "forwarding-sub-layer": {
                "service-outgoing": [
                  {
                    "index": 0,
                    "mpls-label-stack": {
                      "entry": [
                        {
                          "id": 0,
                          "label": 101
                        }
                      ]
                    },
                    "sub-layer": [
                      "fsl-3"
                    ]
                  }
                ]
              }
            }
          },
          {
            "name": "ssl-2",
            "service-rank": 10,
            "traffic-profile": "pf-1",
            "service-protection": {
              "protection": "none",
              "sequence-number-length": "long-sn"
            },
```

```
            "operation": "relay",
            "incoming": {
              "service-id": {
                "mpls-label-stack": {
                  "entry": [
                    {
                      "id": 0,
                      "label": 103
                    }
                  ]
                }
              }
            },
            "outgoing": {
              "forwarding-sub-layer": {
                "service-outgoing": [
                  {
                    "index": 0,
                    "mpls-label-stack": {
                      "entry": [
                        {
                          "id": 0,
                          "label": 104
                        }
                      ]
                    },
                    "sub-layer": [
                      "fsl-4"
                    ]
                  }
                ]
              }
            }
          },
          {
            "name": "asl-1",
            "service-rank": 10,
            "traffic-profile": "pf-2",
            "service-protection": {
              "protection": "replication",
              "sequence-number-length": "long-sn"
            },
            "operation": "initiation",
            "incoming": {
              "forwarding-aggregation": {
                "sub-layer": [
                  "fsl-3",
                  "fsl-4"
```

```
                ]
              }
            },
            "outgoing": {
              "forwarding-sub-layer": {
                "service-outgoing": [
                  {
                    "index": 0,
                    "mpls-label-stack": {
                      "entry": [
                        {
                          "id": 0,
                          "label": 1000
                        }
                      ]
                    },
                    "sub-layer": [
                      "fsl-5",
                      "fsl-6"
                    ]
                  }
                ]
              }
            }
          }
        ]
      },
      "forwarding": {
        "sub-layer": [
          {
            "name": "fsl-1",
            "traffic-profile": "pf-3",
            "operation": "pop-and-lookup",
            "incoming": {
              "forwarding-id": {
                "interface": "eth0",
                "mpls-label-stack": {
                  "entry": [
                    {
                      "id": 0,
                      "label": 10000
                    }
                  ]
                }
              }
            },
            "outgoing": {
              "service-sub-layer": {
```

```
              "sub-layer": [
                "ssl-1"
              ]
            }
          }
        },
        {
          "name": "fsl-2",
          "traffic-profile": "pf-3",
          "operation": "pop-and-lookup",
          "incoming": {
            "forwarding-id": {
              "interface": "eth1",
              "mpls-label-stack": {
                "entry": [
                  {
                    "id": 0,
                    "label": 10006
                  }
                ]
              }
            }
          },
          "outgoing": {
            "service-sub-layer": {
              "sub-layer": [
                "ssl-2"
              ]
            }
          }
        },
        {
          "name": "fsl-3",
          "traffic-profile": "pf-3",
          "operation": "impose-and-forward",
          "incoming": {
            "service-sub-layer": {
              "sub-layer": [
                "ssl-1"
              ]
            }
          },
          "outgoing": {
            "service-aggregation": {
              "aggregation-sub-layer": "asl-1",
              "optional-forwarding-label": {
                "mpls-label-stack": {
                  "entry": [
```

```
                    {
                      "id": 0,
                      "label": 20004
                    }
                  ]
                }
              }
            }
          }
        },
        {
          "name": "fsl-4",
          "traffic-profile": "pf-3",
          "operation": "impose-and-forward",
          "incoming": {
            "service-sub-layer": {
              "sub-layer": [
                "ssl-2"
              ]
            }
          },
          "outgoing": {
            "service-aggregation": {
              "aggregation-sub-layer": "asl-1",
              "optional-forwarding-label": {
                "mpls-label-stack": {
                  "entry": [
                    {
                      "id": 0,
                      "label": 20005
                    }
                  ]
                }
              }
            }
          }
        },
        {
          "name": "fsl-5",
          "traffic-profile": "pf-4",
          "operation": "impose-and-forward",
          "incoming": {
            "service-sub-layer": {
              "sub-layer": [
                "asl-1"
              ]
            }
          },
```

```
            "outgoing": {
              "interface": {
                "outgoing-interface": "eth2",
                "mpls-label-stack": {
                  "entry": [
                    {
                      "id": 0,
                      "label": 20000
                    }
                  ]
                }
              }
            }
          },
          {
            "name": "fsl-6",
            "traffic-profile": "pf-4",
            "operation": "impose-and-forward",
            "incoming": {
              "service-sub-layer": {
                "sub-layer": [
                  "asl-1"
                ]
              }
            },
            "outgoing": {
              "interface": {
                "outgoing-interface": "eth3",
                "mpls-label-stack": {
                  "entry": [
                    {
                      "id": 0,
                      "label": 20001
                    }
                  ]
                }
              }
            }
          }
        ]
      }
    },
    "ietf-interfaces:interfaces": {
      "interface": [
        {
          "name": "eth0",
          "type": "iana-if-type:ethernetCsmacd",
          "oper-status": "up",
```

```
          "statistics": {
            "discontinuity-time": "2024-02-21T18:59:00-05:00"
          }
        },
        {
          "name": "eth1",
          "type": "iana-if-type:ethernetCsmacd",
          "oper-status": "up",
          "statistics": {
            "discontinuity-time": "2024-02-21T18:59:00-05:00"
          }
        },
        {
          "name": "eth2",
          "type": "iana-if-type:ethernetCsmacd",
          "oper-status": "up",
          "statistics": {
            "discontinuity-time": "2024-02-21T18:59:00-05:00"
          }
        },
        {
          "name": "eth3",
          "type": "iana-if-type:ethernetCsmacd",
          "oper-status": "up",
          "statistics": {
            "discontinuity-time": "2024-02-21T18:59:00-05:00"
          }
        }
      ]
    }
  }
```


Figure 19: Example C-4 DetNet Relay Service Sub-Layer Aggregation


```
  {
    "ietf-detnet:detnet": {
      "traffic-profile": [
        {
          "name": "pf-1",
          "traffic-requirements": {
            "min-bandwidth": "100000000",
            "max-latency": 100000000,
            "max-latency-variation": 10000000,
            "max-loss": "0.0000001",
            "max-consecutive-loss-tolerance": 5,
            "max-misordering": 0
          },
          "member-svc-sublayer": [
```

```
            "ssl-1",
            "ssl-2"
          ]
        },
        {
          "name": "pf-2",
          "traffic-requirements": {
            "min-bandwidth": "200000000",
            "max-latency": 100000000,
            "max-latency-variation": 10000000,
            "max-loss": "0.0000001",
            "max-consecutive-loss-tolerance": 5,
            "max-misordering": 0
          },
          "member-svc-sublayer": [
            "asl-1"
          ]
        },
        {
          "name": "pf-3",
          "traffic-spec": {
            "interval": 125,
            "max-pkts-per-interval": 1,
            "max-payload-size": 1518
          },
          "member-fwd-sublayer": [
            "fsl-3",
            "fsl-4",
            "fsl-5",
            "fsl-6"
          ]
        },
        {
          "name": "pf-4",
          "traffic-spec": {
            "interval": 125,
            "max-pkts-per-interval": 2,
            "max-payload-size": 1518
          },
          "member-fwd-sublayer": [
            "fsl-1",
            "fsl-2"
          ]
        }
      ],
      "service": {
        "sub-layer": [
          {
```

```
            "name": "ssl-1",
            "service-rank": 10,
            "traffic-profile": "pf-1",
            "service-protection": {
              "protection": "none",
              "sequence-number-length": "long-sn"
            },
            "operation": "relay",
            "incoming": {
              "service-id": {
                "mpls-label-stack": {
                  "entry": [
                    {
                      "id": 0,
                      "label": 101
                    }
                  ]
                }
              }
            },
            "outgoing": {
              "forwarding-sub-layer": {
                "service-outgoing": [
                  {
                    "index": 0,
                    "mpls-label-stack": {
                      "entry": [
                        {
                          "id": 0,
                          "label": 102
                        }
                      ]
                    },
                    "sub-layer": [
                      "fsl-5"
                    ]
                  }
                ]
              }
            }
          },
          {
            "name": "ssl-2",
            "service-rank": 10,
            "traffic-profile": "pf-1",
            "service-protection": {
              "protection": "none",
              "sequence-number-length": "long-sn"
```

```
            },
            "operation": "relay",
            "incoming": {
              "service-id": {
                "mpls-label-stack": {
                  "entry": [
                    {
                      "id": 0,
                      "label": 104
                    }
                  ]
                }
              }
            },
            "outgoing": {
              "forwarding-sub-layer": {
                "service-outgoing": [
                  {
                    "index": 0,
                    "mpls-label-stack": {
                      "entry": [
                        {
                          "id": 0,
                          "label": 105
                        }
                      ]
                    },
                    "sub-layer": [
                      "fsl-6"
                    ]
                  }
                ]
              }
            }
          },
          {
            "name": "asl-1",
            "service-rank": 10,
            "traffic-profile": "pf-2",
            "service-protection": {
              "protection": "elimination",
              "sequence-number-length": "long-sn"
            },
            "operation": "termination",
            "incoming": {
              "service-id": {
                "mpls-label-stack": {
                  "entry": [
```

```
                    {
                      "id": 0,
                      "label": 1000
                    }
                  ]
                }
              }
            },
            "outgoing": {
              "forwarding-disaggregation": {
                "sub-layer": [
                  "fsl-3",
                  "fsl-4"
                ]
              }
            }
          }
        ]
      },
      "forwarding": {
        "sub-layer": [
          {
            "name": "fsl-1",
            "traffic-profile": "pf-4",
            "operation": "pop-and-lookup",
            "incoming": {
              "forwarding-id": {
                "interface": "eth0",
                "mpls-label-stack": {
                  "entry": [
                    {
                      "id": 0,
                      "label": 20002
                    }
                  ]
                }
              }
            },
            "outgoing": {
              "service-sub-layer": {
                "sub-layer": [
                  "asl-1"
                ]
              }
            }
          },
          {
            "name": "fsl-2",
```

```
            "traffic-profile": "pf-4",
            "operation": "pop-and-lookup",
            "incoming": {
              "forwarding-id": {
                "interface": "eth1",
                "mpls-label-stack": {
                  "entry": [
                    {
                      "id": 0,
                      "label": 20003
                    }
                  ]
                }
              }
            },
            "outgoing": {
              "service-sub-layer": {
                "sub-layer": [
                  "asl-1"
                ]
              }
            }
          },
          {
            "name": "fsl-3",
            "traffic-profile": "pf-3",
            "operation": "pop-and-lookup",
            "incoming": {
              "forwarding-id": {
                "interface": "eth0",
                "mpls-label-stack": {
                  "entry": [
                    {
                      "id": 0,
                      "label": 20004
                    }
                  ]
                }
              }
            },
            "outgoing": {
              "service-sub-layer": {
                "sub-layer": [
                  "ssl-1"
                ]
              }
            }
          },
```

```
          {
            "name": "fsl-4",
            "traffic-profile": "pf-3",
            "operation": "pop-and-lookup",
            "incoming": {
              "forwarding-id": {
                "interface": "eth1",
                "mpls-label-stack": {
                  "entry": [
                    {
                      "id": 0,
                      "label": 20005
                    }
                  ]
                }
              }
            },
            "outgoing": {
              "service-sub-layer": {
                "sub-layer": [
                  "ssl-2"
                ]
              }
            }
          },
          {
            "name": "fsl-5",
            "traffic-profile": "pf-3",
            "operation": "impose-and-forward",
            "incoming": {
              "service-sub-layer": {
                "sub-layer": [
                  "ssl-1"
                ]
              }
            },
            "outgoing": {
              "interface": {
                "outgoing-interface": "eth2",
                "mpls-label-stack": {
                  "entry": [
                    {
                      "id": 0,
                      "label": 10005
                    }
                  ]
                }
              }
```

```
            }
          },
          {
            "name": "fsl-6",
            "traffic-profile": "pf-3",
            "operation": "impose-and-forward",
            "incoming": {
              "service-sub-layer": {
                "sub-layer": [
                  "ssl-2"
                ]
              }
            },
            "outgoing": {
              "interface": {
                "outgoing-interface": "eth3",
                "mpls-label-stack": {
                  "entry": [
                    {
                      "id": 0,
                      "label": 10011
                    }
                  ]
                }
              }
            }
          }
        ]
      }
    },
    "ietf-interfaces:interfaces": {
      "interface": [
        {
          "name": "eth0",
          "type": "iana-if-type:ethernetCsmacd",
          "oper-status": "up",
          "statistics": {
            "discontinuity-time": "2024-02-21T18:59:00-05:00"
          }
        },
        {
          "name": "eth1",
          "type": "iana-if-type:ethernetCsmacd",
          "oper-status": "up",
          "statistics": {
            "discontinuity-time": "2024-02-21T18:59:00-05:00"
          }
        },
```

```
        {
          "name": "eth2",
          "type": "iana-if-type:ethernetCsmacd",
          "oper-status": "up",
          "statistics": {
            "discontinuity-time": "2024-02-21T18:59:00-05:00"
          }
        },
        {
          "name": "eth3",
          "type": "iana-if-type:ethernetCsmacd",
          "oper-status": "up",
          "statistics": {
            "discontinuity-time": "2024-02-21T18:59:00-05:00"
          }
        }
      ]
    }
  }
```

Figure 20: Example C-4 DetNet Relay Service Sub-Layer Disaggregation

B.8. Example D-1 JSON Transit Forwarding Sub-Layer Aggregation/ Disaggregation

Figure 21 illustrates at the transit node, forwarding sub-layer flows 1 and 2 are aggregated into a single forwarding sub-layer. Resource allocation is performed by the corresponding forwarding sub-layer for all aggregated flows. Figure 21 illustrating both aggregation and disaggregation is shown and then the corresponding JSON operational data follows.

Please consult the PDF or HTML versions for the Case D-1 Diagram

Figure 21: Case D-1 Example Transit Node Forwarding Aggregation/ Disaggregation

Figure 22 contains the operational JSON configuration for the ingress aggregation node illustrated in Figure 21. In this example, a transit node performing aggregation at the forwarding sub-layer is illustrated. Two DetNet flows DN1 and DN2 are transmitted at each forwarding sub-layer. The DetNet forwarding sub-layer flows with MPLS labels 10002 and 10006 are aggregated at the single forwarding sub-layer. The resulting aggregated DetNet flow has MPLS label 20000. Figure 23 contains the operational JSON configuration for the egress disaggregation transit node illustrated in Figure 21.

```
{
  "ietf-detnet:detnet": {
    "traffic-profile": [
      {
        "name": "pf-1",
        "traffic-spec": {
          "interval": 125,
          "max-pkts-per-interval": 1,
          "max-payload-size": 1518
        },
        "member-fwd-sublayer": [
          "fsl-1",
          "fsl-2"
        ]
      },
      {
        "name": "pf-2",
        "traffic-spec": {
          "interval": 125,
          "max-pkts-per-interval": 2,
          "max-payload-size": 1518
        },
        "member-fwd-sublayer": [
          "afl-1"
        ]
      }
    ],
    "forwarding": {
      "sub-layer": [
        {
          "name": "fsl-1",
          "traffic-profile": "pf-1",
          "operation": "pop-impose-and-forward",
          "incoming": {
            "forwarding-id": {
              "interface": "eth0",
              "mpls-label-stack": {
                "entry": [
                  {
                    "id": 0,
                    "label": 10000
                  }
                ]
              }
            }
          },
          "outgoing": {
            "forwarding-sub-layer": {
```

```
                "aggregation-sub-layer": "afl-1",
                "forwarding-label": {
                  "mpls-label-stack": {
                    "entry": [
                      {
                        "id": 0,
                        "label": 10002
                      }
                    ]
                  }
                }
              }
            }
          },
          {
            "name": "fsl-2",
            "traffic-profile": "pf-1",
            "operation": "pop-impose-and-forward",
            "incoming": {
              "forwarding-id": {
                "interface": "eth1",
                "mpls-label-stack": {
                  "entry": [
                    {
                      "id": 0,
                      "label": 10004
                    }
                  ]
                }
              }
            },
            "outgoing": {
              "forwarding-sub-layer": {
                "aggregation-sub-layer": "afl-1",
                "forwarding-label": {
                  "mpls-label-stack": {
                    "entry": [
                      {
                        "id": 0,
                        "label": 10006
                      }
                    ]
                  }
                }
              }
            }
          },
          {
```

```
            "name": "af1-1",
            "traffic-profile": "pf-2",
            "operation": "impose-and-forward",
            "incoming": {
              "forwarding-aggregation": {
                "sub-layer": [
                  "fsl-1",
                  "fsl-2"
                ]
              }
            },
            "outgoing": {
              "interface": {
                "outgoing-interface": "eth3",
                "mpls-label-stack": {
                  "entry": [
                    {
                      "id": 0,
                      "label": 20000
                    }
                  ]
                }
              }
            }
          }
        ]
      }
    },
    "ietf-interfaces:interfaces": {
      "interface": [
        {
          "name": "eth0",
          "type": "iana-if-type:ethernetCsmacd",
          "oper-status": "up",
          "statistics": {
            "discontinuity-time": "2024-02-21T18:59:00-05:00"
          }
        },
        {
          "name": "eth1",
          "type": "iana-if-type:ethernetCsmacd",
          "oper-status": "up",
          "statistics": {
            "discontinuity-time": "2024-02-21T18:59:00-05:00"
          }
        },
        {
          "name": "eth3",
```

```
          "type": "iana-if-type:ethernetCsmacd",
          "oper-status": "up",
          "statistics": {
            "discontinuity-time": "2024-02-21T18:59:00-05:00"
          }
        }
      ]
    }
  }
```

Figure 22: Example D-1 Transit Node Forwarding Aggregation

```
  {
    "ietf-detnet:detnet": {
      "traffic-profile": [
        {
          "name": "pf-1",
          "traffic-spec": {
            "interval": 125,
            "max-pkts-per-interval": 1,
            "max-payload-size": 1518
          },
          "member-fwd-sublayer": [
            "fsl-1",
            "fsl-2"
          ]
        },
        {
          "name": "pf-2",
          "traffic-spec": {
            "interval": 125,
            "max-pkts-per-interval": 2,
            "max-payload-size": 1518
          },
          "member-fwd-sublayer": [
            "afl-1"
          ]
        }
      ],
      "forwarding": {
        "sub-layer": [
          {
            "name": "fsl-1",
            "traffic-profile": "pf-1",
            "operation": "swap-and-forward",
            "incoming": {
              "forwarding-id": {
                "interface": "eth1",
```

```
                "mpls-label-stack": {
                  "entry": [
                    {
                      "id": 0,
                      "label": 10002
                    }
                  ]
                }
              }
            },
            "outgoing": {
              "interface": {
                "outgoing-interface": "eth3",
                "mpls-label-stack": {
                  "entry": [
                    {
                      "id": 0,
                      "label": 10003
                    }
                  ]
                }
              }
            }
          },
          {
            "name": "fsl-2",
            "traffic-profile": "pf-1",
            "operation": "swap-and-forward",
            "incoming": {
              "forwarding-id": {
                "interface": "eth1",
                "mpls-label-stack": {
                  "entry": [
                    {
                      "id": 0,
                      "label": 10006
                    }
                  ]
                }
              }
            },
            "outgoing": {
              "interface": {
                "outgoing-interface": "eth2",
                "mpls-label-stack": {
                  "entry": [
                    {
                      "id": 0,
```

```
                    "label": 10007
                  }
                ]
              }
            }
          }
        },
        {
          "name": "afl-1",
          "traffic-profile": "pf-2",
          "operation": "pop-and-lookup",
          "incoming": {
            "forwarding-id": {
              "interface": "eth1",
              "mpls-label-stack": {
                "entry": [
                  {
                    "id": 0,
                    "label": 20001
                  }
                ]
              }
            }
          },
          "outgoing": {
            "forwarding-disaggregation": {
              "sub-layer": [
                "fsl-1",
                "fsl-2"
              ]
            }
          }
        }
      ]
    }
  },
  "ietf-interfaces:interfaces": {
    "interface": [
      {
        "name": "eth1",
        "type": "iana-if-type:ethernetCsmacd",
        "oper-status": "up",
        "statistics": {
          "discontinuity-time": "2024-02-21T18:59:00-05:00"
        }
      },
      {
        "name": "eth2",
```

```
          "type": "iana-if-type:ethernetCsmacd",
          "oper-status": "up",
          "statistics": {
            "discontinuity-time": "2024-02-21T18:59:00-05:00"
          }
        },
        {
          "name": "eth3",
          "type": "iana-if-type:ethernetCsmacd",
          "oper-status": "up",
          "statistics": {
            "discontinuity-time": "2024-02-21T18:59:00-05:00"
          }
        }
      ]
    }
  }
```

Figure 23: Example D-1 Transit Node Forwarding Disaggregation

Authors' Addresses

Xuesong Geng
Huawei Technologies
Email: gengxuesong@huawei.com

Yeoncheol Ryoo
ETRI
Email: dbduscjf@etri.re.kr

Don Fedyk
LabN Consulting, L.L.C.
Email: dfedyk@labn.net

Reshad Rahman
Equinix
Email: reshad@yahoo.com

Zhenqiang Li
China Mobile
Email: lizhenqiang@chinamobile.com

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

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

October 25, 2020

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

Abstract

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

Status of This Memo

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

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

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

This Internet-Draft will expire on April 28, 2021.

Copyright Notice

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

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

Table of Contents

1. TEMPORARY EDITORIAL NOTES 2
2. Introduction 3
2.1. Terminology 3
2.2. Acronyms 4
2.3. Requirements Language 4
3. Role of OAM in DetNet 4
4. Operation 5
4.1. Information Collection 5
4.2. Continuity Check 5
4.3. Connectivity Verification 6
4.4. Route Tracing 6
4.5. Fault Verification/detection 6
4.6. Fault Isolation/identification 7
5. Administration 7
5.1. Collection of metrics 7
5.2. Worst-case metrics 7
6. Maintenance 8
6.1. Replication / Elimination 8
6.2. Resource Reservation 8
6.3. Soft transition after reconfiguration 9
7. IANA Considerations 9
8. Security Considerations 9
9. Acknowledgments 9
10. Informative References 9
Authors' Addresses 10

1. TEMPORARY EDITORIAL NOTES

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

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

2. Introduction

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

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

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

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

2.1. Terminology

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

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

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

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

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

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

- 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.

2.2. Acronyms

OAM: Operations, Administration, and Maintenance

DetNet: Deterministic Networking

SLO: Service Level Objective

QoS: Quality of Service

SNMP: Simple Network Management Protocol

SDN: Software Defined Network

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

2.3. Requirements Language

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

3. Role of OAM in DetNet

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

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

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

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

4. Operation

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

4.1. Information Collection

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

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

4.2. Continuity Check

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

4.3. Connectivity Verification

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

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

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

4.4. Route Tracing

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

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

4.5. Fault Verification/detection

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

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

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

4.6. Fault Isolation/identification

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

5. Administration

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

- Queuing Delay: the time elapsed between a packet enqueued and its transmission to the next hop.
- Buffer occupancy: the number of packets present in the buffer, for each of the existing flows.

The following metrics SHOULD be collected:

- per virtual circuit to measure the end-to-end performance for a given flow. Each of the paths has to be isolated in multipath routing strategies.
- per path to detect misbehaving path when multiple paths are applied.
- per device to detect misbehaving node, when it relays the packets of several flows.

5.1. Collection of metrics

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

5.2. Worst-case metrics

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

6. Maintenance

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

the cost of the sub-optimality: resources may not be used optimally (e.g., a better path exists).

the reconfiguration cost: the controller needs to trigger some reconfigurations. For this transient period, resources may be twice reserved, and control packets have to be transmitted.

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

6.1. Replication / Elimination

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

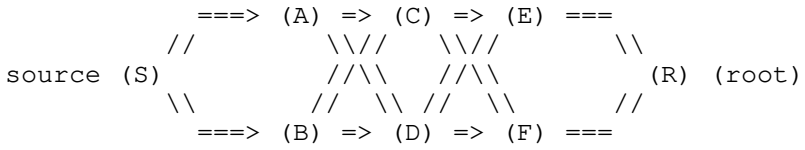


Figure 1: Packet Replication: S transmits twice the same data packet, to DP(A) and AP (B).

6.2. Resource Reservation

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

6.3. Soft transition after reconfiguration

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

7. IANA Considerations

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

8. Security Considerations

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

9. Acknowledgments

TBD

10. Informative References

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

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

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

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

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

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

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

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

Authors' Addresses

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

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

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

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

Grek Mirsky
ZTE Corp.

Email: gregimirsky@gmail.com

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

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

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

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

October 23, 2020

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

Abstract

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

Status of this Memo

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

Copyright and License Notice

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

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

Table of Contents

1 Introduction 3
1.1 Terminology 3
2 Distributed Control in Bridged TSN-based Ethernet Deployment . 3
2.1 Overview 3
2.2 RAP Reservation in TSN vs RSVP IntServ Model 4
2.3 Interactions between L2 and L3 5
2.4 Similarities and Differences between RSVP and RAP 6
2.5. RSVP-DetNet 8
2.6. API Specifications 9
3. Centralized Control Signaling in SDN Domain 16
4. Hybrid Control Signaling in SDN Domain 16
5. Security Considerations 16
6. IANA Considerations 16
7. Conclusion 16
8. References 16
8.1 Normative References 17
8.2 Informative References 17
Authors' Addresses 17

1 Introduction

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

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

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

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

1.1 Terminology

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

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

2 Distributed Control in Bridged TSN-based Ethernet Deployment

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

2.1 Overview

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

for reservation of aggregated flows within the core network.

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

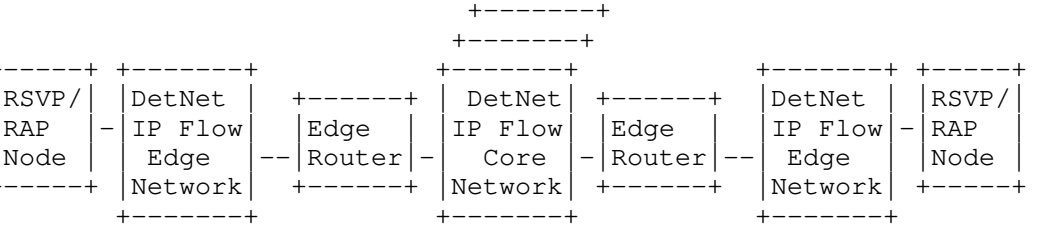


Figure 1 : IP + Bridged Ethernet Within Customer Networks

2.2 RAP Reservation in TSN vs RSVP IntServ Model

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

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

This difference in models poses a number of challenges:

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

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

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

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

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

2.3 Interactions between L2 and L3

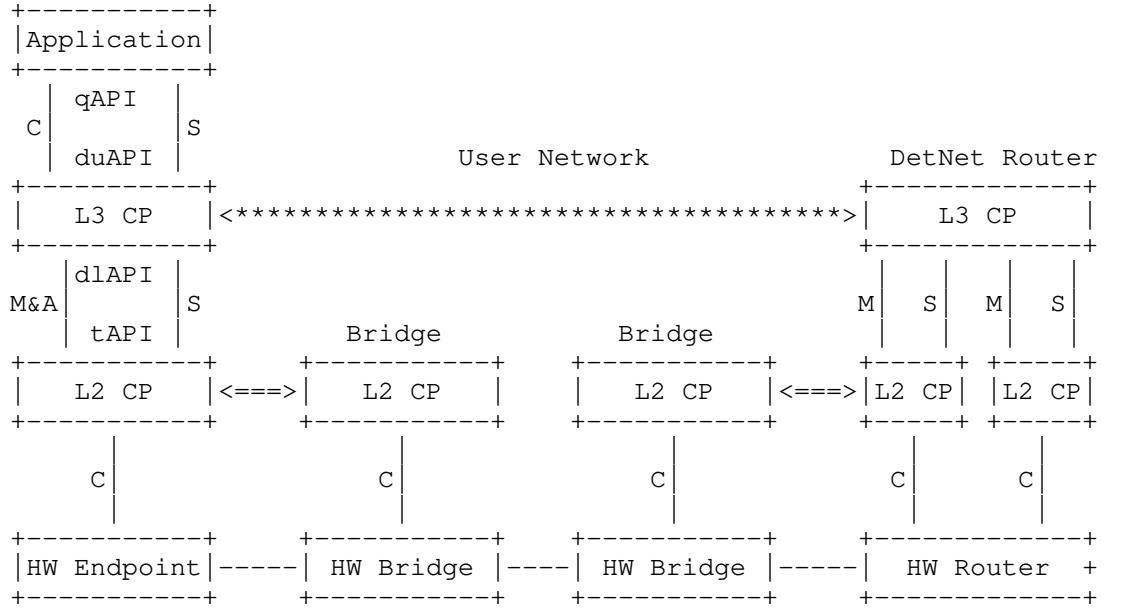
Figure 2 provides an overview of the interactions between L2 and L3 elements in a network deployment as an elaboration of the elements in Figure 1.

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

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

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

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



```
***** L3 Signaling Control Protocol     DetNet UNI (dUNI RSVP)
===== L2 Reservation Control Protocol TSN UNI (tUNI IEEE P802.1Qdd)

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

Figure 2 : Interaction between L2 and L3

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

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

2.4 Similarities and Differences between RSVP and RAP

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

2.4.1 Assumptions on Network Nodes

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

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

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

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

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

- All nodes take part in the signaling process

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

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

2.4.2 Mapping of Latency Model

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

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

2.4.3 Dealing with Latency Margins

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

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

2.4.4 Dealing with Jitter and Non-Shaping Nodes

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

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

neighbor.

2.4.5 Mapping Resource Reservation Styles

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

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

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

Figure 3 : Resource Style and Sender Selection [RFC2205]

2.5. RSVP-DetNet

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

2.5.1. Rationale

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

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

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

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

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

2.5.2. Splitting Control over Resource Style and Sender Selection

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

2.5.3. Coordinated Shared Resource Style

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

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

2.6. API Specifications

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

2.6.1. RSVP-DetNet upper API (duAPI)

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

Sender

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

Request parameter:

- Flow destination IP address, Protocol ID, Destination Port

Response parameter:

- L3 Session ID (local handle)

Call: Add IP Flow

Request parameter

- L3 Session ID, Sender source IP address, Source Port, DSCP

- Traffic Specification: Maximum IP packet size (per flow, <= MTU), Minimum IP packet size, Burstiness, IP packet information rate

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

- Data TTL, PATH MTU size, Loss Rate

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

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

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

The Loss Rate is accumulated per IP Flow.

Upcall: IP Flow

- L3 Session ID

- One of the Info_type: RESV_EVENT; PATH_ERROR

Receiver

Call: Open L3 Session

Request parameter

- Flow destination IP address, Protocol ID, Destination Port

Response parameter

- L3 Session ID

Call: Attach IP Flow

Request parameter

- L3 Session ID

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

- Maximum packet size

- Extended Traffic Specification: Maximum Expected Latency

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

Upcall: IP Flow

- L3 Session ID

- One of the Info_type: RESV_EVENT; PATH_ERROR

General

Call: Close L3 Session

Request parameter

- L3 Session ID

2.6.2. RSVP-DetNet lower API (dlAPI)

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

Sender

Call: Add Flow

Request parameter

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

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

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

Response parameter

- Transport Flow Identification

Notes for new parameter:

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

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

Upcall: Flow

Response parameter

- L3 Session ID, Transport Flow Identification

- One of the Info_type: RESV_EVENT, RES_MODIFY_EVENT

Receiver

Call: Attach Flow

Request parameter

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

- Extended Traffic Specification: Maximum expected latency

- One of the Info_type: ANNOUNCE_EVENT, ANNOUNCE_MODIFY_EVENT

Notes for new parameter:

(see notes above)

Upcall: Flow

Response parameter

- L3 Session ID, Transport Flow Identification

- One of the Info_type: ANNOUNCE_EVENT, ANNOUNCE_MODIFY_EVENT

2.6.3. Example tAPI on RAP defined by IEEE 802.1Q

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

Call: Open L2 Reservation Group

Request parameter

- Stream destination MAC address (unicast or multicast

- VLAN

- PCP

Response parameter

Low-Layer Group ID (local handle)

Notes:

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

Call: Close L2 Reservation Group

Request parameter

- Low-Layer Group ID

Talker

Call: Add Stream

Request parameter

- Low-Layer Group ID

- Stream Identification

- Traffic Specification Template

- Resource Style (Distinct, Shared, Coordinated Shared)

- End-System/End-Station source MAC addres

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

- Sync Domain ID (extension for Coordinated Shared)

- PATH Max frame size

- Talker Loss Rate

Notes:

Traffic Specification Template is queue drain algorithm dependent

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

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

PATH MTU frame size is determined downstream from Talker to Listener

The Loss Rate is accumulated per Stream.

Call: Modify Stream

Request parameter

- Low-Layer Group ID

- Stream Identification,

- Traffic Specification Template

Call: Release Stream

Request parameter

- Low-Layer Group ID

- Stream Identification

Upcall: Stream

Response parameter

- Low-Layer Group ID, Stream Identification

- One of the Info_type: RESV_EVENT, RESV_MODIFY_EVENT

Listener

Call: Attach Stream

Request parameter

- Low-Layer Group ID,

- Stream Identification

- Maximum frame size

- Schedule Specification (extension for Coordinated Shared)

- Extended Traffic Specification: Maximum expected latency

Notes:

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

Maximum frame size will be delivered upstream from Listener to Talker

Call: Attach Modify Stream

Request parameter

- Low-Layer Group ID

- Stream Identification

- Schedule Specification (extension only for Coordinated Shared)

Call: Release Stream

Request parameter

- Low-Layer Group ID
- Stream Identification

Upcall: Stream

Response parameter

- Low-Layer Group ID
- Stream Identification
- One of the Info_type: ANNOUNCE_EVENT, ANNOUNCE_MODIFY_EVENT

3. Centralized Control Signaling in SDN Domain

For future work.

4. Hybrid Control Signaling in SDN Domain

For future work.

5. Security Considerations

Editor's note: This section needs more details.

6. IANA Considerations

N/A

7. Conclusion

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

8. References

8.1 Normative References

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

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

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

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

8.2 Informative References

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

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

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

Authors' Addresses

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

Email: Dirk.Trossen@Huawei.com

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

Email: franz-josef.goetz@siemens.com

Juergen Schmitt
Siemens AG
Gleiwitzer Str. 555
90475 Nuremberg
Germany

Email: juergen.jues.schmitt@siemens.com

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

Copyright and License Notice

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

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

Table of Contents

1. Introduction 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.

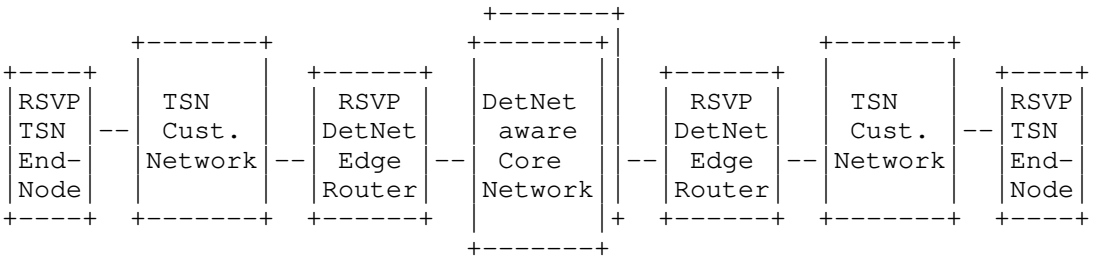


Figure 1 : Distributed DetNet UNI

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.

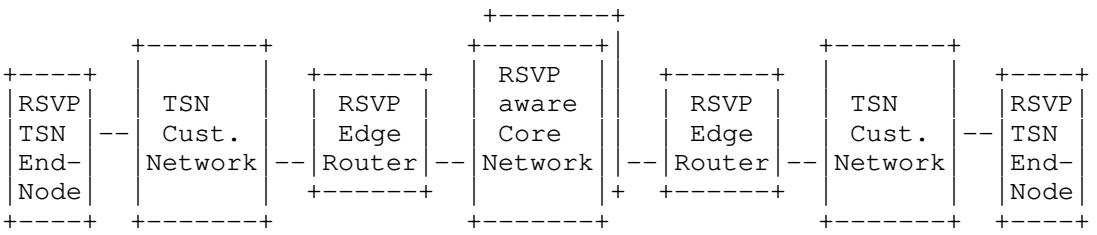


Figure 2 : Fully Distributed DetNet UNI

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.

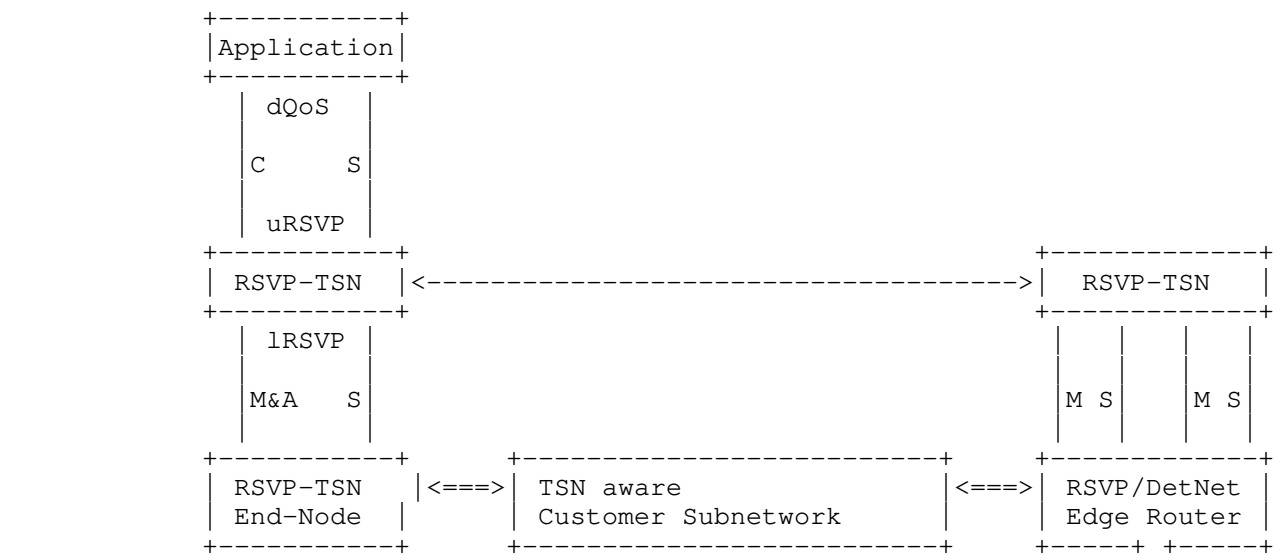


Figure 4: Layer Interactions between RSVP and TSN

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